UNIVERSITÀ DEGLI STUDI DI FIRENZE
FACOLTÀ DI INGEGNERIA

Corso di Sistemi Operativi - Prof. Stefano Berretti

SEMINARIO SULLA VIRTUALIZZAZIONE

a cura di Nicola Fusari

A.A. 2012/2013

1 - INTRODUZIONE

In questi ultimi anni, il tema della *virtualizzazione* è salito (o come vedremo meglio di seguito, *tornato*) prepotentemente alla ribalta nel contesto di moltissime realtà attinenti l'ICT (Information and Communication Technology).
Con il termine *virtualizzazione,* in realtà, si possono indicare contesti, approcci e soluzioni anche molto differenti fra loro, aventi però tutti come **denominatore comune** e base di partenza il **concetto di *astrazione dalla fisicità* delle strutture (dell'hardware) sulle quali si appoggiano**.

Uno degli scenari più tipici a cui si pensa quando si parla di virtualizzazione è quello di più *macchine virtuali* in contemporanea esecuzione su di un unico server fisico (il cosiddetto *host*); in questo contesto (che è quello su cui solitamente i tecnici insistono più volentieri con gli amministratori delegati per convincerli ad approvare i necessari investimenti) la virtualizzazione è ***consolidamento* dell'hardware** cioè un modo per sfruttare al massimo ed il più efficientemente possibile le risorse fisiche, ovvero gli investimenti economici.
Questa è però soltanto una delle numerose sfaccettature della questione: *virtualizzazione*, infatti, è anche la possibilità di presentare più risorse fisiche come un'unica risorsa che possiede -virtualmente, appunto- caratteristiche (di potenza, velocità, capienza, affidabilità, ecc.) decisamente migliori di quelle che caratterizzano le singole realtà fisiche sulle quali si appoggia: questa è ad esempio l'idea alla base del sistema RAID (*Redundant Array of Independent Disks*) di gestione dei dischi, che a seconda della tipologia scelta permette di “vedere” «n» hard disk fisici come un unico disco con capienza, oppure velocità, o ancora affidabilità, moltiplicata per «n» volte rispetto a quella posseduta dalla singola unità fisica; in questa seconda chiave, quindi, la *virtualizzazione* rappresenta la possibilità di **superare i limiti** (che possono anche essere imposti dallo stato dell'arte delle tecnologie impiegate e/o disponibili) **che affliggono l'hardware o più in generale le risorse fisiche** (inutile dire che questo è invece l'aspetto più interessante ed affascinante per i tecnici e per gli AD più “illuminati”).

C'è poi un terzo aspetto che ha contribuito pesantemente all'affermazione delle architetture virtualizzate: oltre ad essere, come accennato con gli esempi precedenti, una soluzione per la razionalizzazione (consolidamento) delle risorse, ovvero per ottenere prestazioni superiori a quelle che si otterrebbero senza la sua introduzione, ed oltre a centrare spesso in buona misura e contemporaneamente entrambi gli obiettivi (che sono poi spesso meno antitetici ed incompatibili di quanto si potrebbe pensare), la *virtualizzazione* apre anche la strada a nuove possibilità per l'***affidabilità* e la *disponibilità*** dei sistemi cui si applica. Proprio il separare la funzionalità dei sistemi ICT dall'hardware sottostante (che è appunto il concetto alla base della virtualizzazione stessa) permette infatti un approccio totalmente nuovo nel fronteggiare guasti, imprevisti o semplicemente fermi programmati per aggiornamenti e/o interventi di manutenzione dei sistemi. In un contesto virtualizzato è infatti possibile, come vedremo meglio, spostare (o come si dice *migrare*) una macchina virtuale (brevemente nel seguito VM, da *virtual machine*) da un server fisico (o come lo chiameremo meglio un *host*) ad un altro, spesso senza che vi sia nemmeno un'interruzione apprezzabile nel funzionamento della VM stessa. Analogamente sarà possibile eseguire backup di un'intera macchina virtuale, così da poterla in qualsiasi momento (ad esempio a seguito di un guasto anche grave dell'hardware su cui si appoggia) ripristinare anche in luoghi e su hardware fisicamente diversi, ottenendo così rapidamente la ripresa delle funzionalità e dei servizi svolti dalla macchina stessa; questo, come si intuisce, apre nuove possibilità e fornisce nuove e più efficaci armi alle moderne strategie di disaster recovery.

Quelli sin qui accennati sono gli argomenti tipici di quello che potremmo definire l'*approccio classico* alla virtualizzazione, ammesso che si possa utilizzare l'aggettivo 'classico' per una tecnologia tutto sommato recente (ma vedremo che per certi aspetti lo è meno di quanto si potrebbe pensare) come quella che attiene appunto i software di virtualizzazione.

Con tale approccio la Virtualizzazione sembrerebbe porsi come la naturale risposta alle esigenze di **realtà aziendali, industriali, scientifiche o di altra**

natura ma comunque tendenzialmente medio-grandi; in tali contesti, infatti, il raggiungimento di un livello il più elevato possibile in termini di prestazioni, affidabilità, disponibilità, (per non parlare del contenimento di consumi ed ingombri tipicamente significativi in realtà di dimensioni importanti), è un aspetto irrinunciabile ed in grado spesso di fare la differenza fra un'esperienza di successo ed una destinata al fallimento; partendo da questo assunto la virtualizzazione si propone come una soluzione ideale non solo per ottenere gli standard necessari, ma anche per ottenerli in modo -se non economico in assoluto- sicuramente *più economico* di quanto non si potrebbe fare con soluzioni 'fisiche' basate sul puro hardware.

D'altro canto, per le **realtà medio piccole** (per le quali ad esempio qualche ora o addirittura giorno di fermo tecnico per quanto seccante e non auspicabile non comporta comunque in generale gravi perdite economiche) potrebbe sembrare -da quanto sin qui detto- che la Virtualizzazione potesse essere liquidata come il classico “cannone per sparare al moscerino”, apparendo magari eccessivi e ingiustificati gli investimenti e le complicazioni tecniche (almeno nella fase di progettazione ed implementazione iniziale della piattaforma stessa) inevitabilmente maggiori della tipica soluzione basata esclusivamente su hardware “fisico”.

In realtà, oltre a quello che abbiamo definito ***'approccio classico'*** alla virtualizzazione (che rimane oggi come oggi innegabilmente quello più diffuso) è possibile -grazie anche ai recenti e continui sviluppi di soluzioni di virtualizzazione derivanti dal mondo open source- adottare anche ***un approccio diverso***, nel contesto del quale la virtualizzazione, pur non rinunciando completamente ai vantaggi sopra descritti, trova impiego anche in realtà piccole o addirittura piccolissime, con soluzioni per contro decisamente economiche; con questo approccio, -del quale cercheremo di vedere rapidamente qualche esempio pratico nella parte finale di questa chiacchierata-, accanto ai concetti di *potenza*, *efficienza*, *affidabilità* e *disponibilità* (un po' meno vitali come si diceva rispetto al

caso delle grandi realtà, ma certo non completamente accantonabili anche per realtà medio-piccole) trovano un posto di rilievo altri aspetti quali ad esempio la possibilità di **isolare** -grazie all'utilizzo di macchine virtuali- **attività potenzialmente pericolose** (informaticamente parlando) per la rete aziendale, o anche la possibilità di **mantenere contemporaneamente** in esercizio sulla stessa macchina fisica **software che richiedono** magari **diverse versioni del sistema operativo** (o addirittura diversi sistemi operativi), o ancora di valutare programmi in **ambienti di prova separati** da quelli di produzione, senza la necessità di acquistare hardware dedicato ai test; non solo: una volta superato il paradigma che vorrebbe la virtualizzazione relegata esclusivamente all'ambito dei server e dei processi più "alti" che avvengono in queste macchine, diventa quasi naturale aprire gli occhi su tutt'altri scenari, nei quali la **virtualizzazione** arriva addirittura **a livello di desktop** e di utente, trasferendo magari alcuni dei concetti cui si accennava sopra, ad esempio la *disponibilità*, da quelle che sono le risorse centrali (come i database aziendali, o i sevizi di autenticazione o ancora quelli di posta elettronica, ecc.) fino a quelle che sono le risorse del singolo utente: questo è ad esempio il caso della cosiddetta ***virtualizzazione del desktop***, con la quale le applicazioni e le impostazioni tipiche che costituiscono l'ambiente di lavoro del singolo operatore, vengono virtualizzate, ovvero, appunto, slegate da un particolare hardware di riferimento, e rese immediatamente disponibili nelle solite forme anche nel caso in cui -ad es. per un guasto o semplicemente per un momentaneo trasferimento dell'operatore ad altra postazione- si debba utilizzare un hardware fisicamente diverso dal solito.

Molti di questi aspetti, qui appena accennati per ragioni di tempo, costituiscono in buona sostanza l'oggetto di quella che potremmo chiamare una ***virtualizzazione a misura di piccola impresa***.

2.1 - RACCORDI CON IL CORSO DI S.O.

Prima di entrare nel vivo di questo seminario sulla virtualizzazione, ed al fine di meglio collocarlo all'interno del Corso di S.O. è utile richiamare rapidamente alcuni concetti fondamentali riguardanti i S.O. così come presentati ad esempio dal testo di Silberschatz (Sistemi Operativi - Concetti ed Esempi). [1]

Definizione di Sistema Operativo

Sebbene una definizione univoca ed esaustiva di *Sistema Operativo* risulti (così come per il termine *Virtualizzazione*) abbastanza difficile da individuare, possiamo senz'altro dire che il S.O. è uno dei quattro componenti fondamentali di un sistema di calcolo, tipicamente scomponibile, appunto, in ***hardware***, ***sistema operativo***, ***applicativi*** e ***utenti***.

Più in particolare possiamo dire che ***il S.O. è l'ambiente che funge da tramite fra le esigenze dell'utente -che si attuano tramite i vari programmi applicativi- e le risorse del sistema di calcolo*** (*hardware* e *dati*): esso non compie in sostanza alcuna operazione *"utile" di per sé*, ma è il contesto necessario nel quale i vari applicativi possono lavorare in modo utile.

Per far questo esso svolge quindi di volta in volta la funzione di ***allocatore di risorse*** (ad es.: tempo CPU, spazio di memoria, sistemi di I/O, ecc.), oppure di ***programma di controllo*** dell'esecuzione degli applicativi utente con il compito di *evitare errori e conflitti degli stessi fra loro e nei confronti dei vari dispositivi di I/O* (oltre che con il S.O. stesso).

Una definizione generica, ma anche generale e comunemente accettata, di S.O. può essere, in ultima analisi, quella secondo la quale ***un S.O. è l'unico programma che è sempre in esecuzione su un computer*** (e che proprio per questo è detto anche ***kernel***, cioè *nucleo*) a differenza di tutti gli altri applicativi (che rispondono di volta in volta alle particolari esigenze degli utenti).

2.2 - CLASSIFICAZIONE DEI S.O.

Un buon metodo per introdurre una ***classificazione dei sistemi operativi*** può

essere quello di ***ripercorrere rapidamente la loro evoluzione negli ultimi 40 anni*** circa. Tale evoluzione è strettamente legata a quella dell'hardware dei sistemi di calcolo per i quali erano pensati, in un ***rapporto decisamente bidirezionale***: se infatti è ovvio che le caratteristiche hardware dei sistemi di calcolo determinino certe caratteristiche dei rispettivi S.O., è anche vero che *spesso problematiche emerse con l'evoluzione di certe funzionalità dei S.O. hanno storicamente portato all'introduzione di nuove caratteristiche dell'hardware* di generazioni successive.

Classificazione dei S.O.

I primi computer (anni '50 - '60) erano macchine enormi (e assai costose) il cui funzionamento era regolato da una consolle: i dispositivi di ***input*** erano tipicamente lettori di schede perforate o nastri magnetici, mentre quelli di ***output*** erano stampanti, perforatori di schede o ancora nastri magnetici.
In tale contesto l'utente non interagiva direttamente col sistema, bensì tramite i cosiddetti ***job*** (costituiti dal programma e dai dati da processare) che affidava, tipicamente sotto forma di schede perforate, all'operatore del computer.
L'operatore ordinava tipicamente i vari *job* degli utenti-programmatori in *lotti* (o ***batch***) con requisiti analoghi (per ottimizzare i tempi di utilizzo del computer stesso) e li mandava in esecuzione; l'esito di ogni job (dopo un tempo che poteva andare dai minuti ai giorni di elaborazione) veniva poi restituito al relativo programmatore.
Il Sistema Operativo di tali primi computer era piuttosto semplice (si parla appunto di ***S.O. batch semplici***) ed il suo compito era sostanzialmente quello di ricevere in input i batch contenenti le sequenze di job (ad es. tramite un lettore di schede) stampandone tipicamente l'output a fine elaborazione e trasferendo successivamente il controllo al job successivo.
Caratteristica evidente di tali *sistemi batch semplici* era l'***assenza di interazione tra utente e job*** durante l'esecuzione dello stesso ed anche il fatto che la ***CPU del sistema era spesso inattiva***, essendo i tempi dei dispositivi elettromeccanici di I/O sempre decisamente più lenti di quelli di elaborazione della CPU.

Una significativa evoluzione dei S.O. avvenne con l'***introduzione dei dischi come***

supporto di memoria di massa; questo portò due fondamentali novità.

La prima fu che le operazioni di input e di output invece che avvenire con tempi dettati dai lettori di schede e dalle stampanti potevano adesso usare il disco (assai più veloce) come una sorta di ***buffer per i dati***. Questo tipo di gestione, chiamato ***spooling*** (acronimo di *simultaneous peripheral operation on-line*) permetteva di fatto di ***sovrapporre temporalmente le operazioni di un job sulle periferiche fisiche di I/O*** (lettori di schede e stampanti) ***con l'esecuzione di altri job*** (che interagivano in prima battuta con il disco), determinando così una significativa *diminuzione del tempo di inattività* della CPU.

La seconda fu che i job non dovevano più essere letti sequenzialmente tramite le schede perforate o i nastri magnetici, ma potevano essere tutti prememorizzati su disco (costituendo quello che si definisce un *pool di job*) ed ***il S.O. operativo poteva eseguire*** il cosiddetto ***scheduling dei job***, decidendo eventualmente l'ordine di accesso agli stessi in modo da minimizzare i tempi 'morti' di attesa (tipicamente di processi lenti quali quelli di I/O) ***riducendo così ulteriormente i tempi di inattività della CPU***.

Con tali capacità aggiuntive (*spooling* e, soprattutto, *scheduling dei job*) si arriva ai cosiddetti ***S.O. batch multiprogrammati***: questi sono i primi S.O. che devono *"prendere decisioni per l'utente"* potendo scegliere l'ordine dei job e quindi la priorità di esecuzione degli stessi, cominciando anche ad assumere quella connotazione di *"allocatori di risorse"* condivise fra i vari job di cui si diceva sopra.

L'evoluzione successiva dei S.O. è legata ancora una volta anche ad un'evoluzione hardware dei sistemi di calcolo. Pian piano l'esigenza di interazione fra utente-programmatore e S.O. diventò sempre più sentita, i lettori di schede vennero soppiantati dalle tastiere (e mouse) e le stampanti dagli schermi.

A quel punto, quasi come naturale estensione logica della *multiprogrammazione* si arrivò a ***S.O.*** che impiegavano il ***time-sharing*** ovvero la ***commutazione della CPU fra l'esecuzione di più job*** con una frequenza tale da permettere agli utenti (ed anche a più utenti contemporaneamente) di interagire con i propri programmi;

poiché le interazioni degli utenti con il sistema avvenivano con tempi "umani" comunque assai più lunghi rispetto a quelli di elaborazione (anche con le CPU dell'epoca!) ogni utente aveva l'impressione di lavorare da solo sul computer mentre in effetti questo era condiviso da molti utenti.

I S.O. time-sharing riuscivano così ad ***ottimizzare ulteriormente l'impiego del tempo macchina della CPU***, massimizzando il ritorno (in termini anche di numero di utilizzatori contemporanei) dei grandi investimenti necessari per i costosi mainframe dell'epoca.

Ovviamente ***il passaggio a S.O. time-sharing accentuò quell'aumento di complessità*** che si era già avuto nel passare dai *sistemi batch semplici* a quelli *multiprogrammati*: oltre alla funzione di ***allocatore di risorse*** (sempre più importante e vitale) il S.O. assume sempre di più anche la funzione di ***controllore*** con il compito di ***evitare interazioni indesiderate e conflitti*** nell'accesso alle risorse disponibili da parte dei vari job in contemporanea esecuzione e dei vari utenti che adesso interagiscono con essi.

Con il passare degli anni (tra gli anni '70 e gli anni '80 del secolo scorso) l'abbassamento dei costi dell'hardware (con l'avvento dei personal computer) rese possibile tornare a pensare di dedicare un solo sistema di calcolo ad un singolo utente.

I ***S.O. operativi pensati per i personal computer*** subirono una nuova evoluzione dettata dal fatto che, in virtù dell'abbattimento dei costi e della relativa diffusione dei PC, *l'obiettivo della massimizzazione dell'utilizzo della CPU non era più quello predominante*, ma veniva *affiancato e addirittura superato da quello di rispondere a requisiti di **praticità e prontezza d'uso** per gli utenti*.

Inizialmente sembrò che molte funzioni evolute adottate per i S.O. operativi dei mainframe non fossero necessarie (o risultassero addirittura inadatte) per quelli dei personal computer.

In realtà ben presto sia per l'aumento della potenza dell'hardware in dotazione ai PC sia per il complicarsi delle interazioni degli utenti con questi (ed anche dei PC, sempre più connessi in rete, fra loro) ***molte delle caratteristiche appannaggio dei***

S.O. per mainframe migrarono progressivamente all'interno dei moderni Sistemi Operativi per personal computer (solo per fare un esempio il *multitasking* è evidentemente figlio della *multiprogrammazione* e del *time-sharing*).

Per completare questa rapida classificazione dei Sistemi di Calcolo (e dei relativi S.O.) vogliamo infine solo accennare anche ai ***Sistemi Paralleli***, caratterizzati da più CPU in stretta comunicazione e che condividono bus, memoria e dispositivi periferici, ed ai ***Sistemi Distribuiti***, nei quali invece più processori -ciascuno con le proprie risorse- possono essere dislocati in sedi remote e sono in collegamento (tra loro e con gli utenti) tramite una apposita rete.
Infine un discorso a parte riguarda i cosiddetti ***Sistemi real-time*** utilizzati in tutte quelle situazioni dove sono richiesti **vincoli di tempo più (*hard real-time*) o meno (*soft real-time*) rigidi**, come ad esempio in tutti quei sistemi di calcolo alla base di sistemi di controllo automatico in apparecchiature di tipo scientifico, biomedicale o industriale.

2.3 - ALCUNI MECCANISMI DI FUNZIONAMENO DEI SISTEMI DI CALCOLO - CENNI

A molti degli aspetti dei S.O. discussi nella breve classificazione fatta sopra, sono legati, come già detto, svariati meccanismi di funzionamento (e anche vere e proprie caratteristiche hardware) dei sistemi di calcolo.
In questo seminario, per brevità, non tratteremo gran parte di tali meccanismi (peraltro probabilmente già discussi a fondo in altre parti del corso di Sistemi Operativi).
Alcuni di essi, però, sono particolarmente importanti nell'ambito della Virtualizzazione, per cui cercheremo di spendere su di esse qualche parola in più.

Come abbiamo visto i sistemi multiprogrammati e time-sharing migliorano le prestazioni sovrapponendo le operazioni di CPU e di I/O (anche di job diversi) sulla medesima macchina; questa sovrapposizione temporale richiede un'attenta

gestione da parte del S.O. del trasferimento dati tra CPU e sistemi di I/O; i meccanismi per realizzare questa gestione sono l'interrogazione ciclica (***polling***), l'utilizzo degli ***interrupt*** oppure dell'accesso diretto alla memoria (***DMA***) e non ci addentreremo nella loro trattazione.
Sempre come conseguenza della *multiprogrammazione* abbiamo anche visto che generalmente ci sono ***più dati e programmi (relativi a job differenti) contemporaneamente residenti nella memoria*** centrale. Per effetto del *time-sharing*, inoltre, questi programmi saranno ***contemporaneamente in esecuzione*** nello stesso sistema rispondendo tipicamente alle interazioni/richieste provenienti da diversi utenti.
Il Sistema Operativo deve assicurare il corretto funzionamento del computer, in sinergia con particolari mezzi che l'hardware su cui gira (e per cui è progettato) gli mette a disposizione.
Per ***evitare che i programmi utenti vadano ad interferire fra loro e con il codice del S.O.*** (che come questi risiede nella medesima memoria centrale) la CPU ha ***due distinte modalità di funzionamento: la modalità utente e la modalità monitor*** (detta anche spesso *modalità privilegiata* o *supervisore*).
Il duplice modo di funzionamento consente sia di proteggere il sistema operativo dal comportamento degli applicativi-utente (quindi, potremmo dire, dagli utenti) che di proteggere gli utenti dagli altri utenti.
Questo si ottiene definendo le istruzioni macchina in grado di causare danni allo stato del sistema come ***istruzioni privilegiate***. Poiché l'hardware consente l'esecuzione di queste istruzioni soltanto in modalità monitor, ***se si tenta di eseguire in modo utente un'istruzione privilegiata la CPU non la esegue*** perché l'hardware la tratta come istruzione illegale generando un'eccezione (*trap*) che rimanda al Sistema Operativo.
Le ***istruzioni per la modifica dei registri di controllo della memoria*** e le ***istruzioni per il controllo dell'I/O*** sono esempi di ***istruzioni privilegiate***. Un'altra istruzione privilegiata è ad esempio l'istruzione *halt*, visto che un programma utente non deve avere la possibilità di provocare l'arresto di tutti gli altri programmi (né tantomeno del S.O.).

Ecco quindi che ***per eseguire ad esempio un'operazione di I/O*** (così come una qualsiasi istruzione privilegiata) ***un programma utente deve sempre necessariamente richiedere al S.O. di farlo in sua vece*** (con quella che si definisce una *system call* cioè con un'operazione che genera un interrupt appunto verso il S.O.). Il Sistema Operativo, lavorando in modalità monitor, ***verifica la validità della richiesta*** (e la compatibilità con la propria integrità e con quella di tutti gli altri applicativi utente!) ***ed in caso affermativo la soddisfa***, restituendo poi il controllo al programma utente.

La presenza di diversi livelli di privilegio per l'esecuzione di determinate istruzioni (o per l'accesso a determinate porzioni di memoria) verrà ripresa fra breve quando parleremo degli anelli (ring) di privilegio e della collocazione rispetto ad essi dei software di Virtualizzazione.

3 - LE BASI DELLA VIRTUALIZZAZIONE

Come già detto introducendo questo Seminario, il termine *virtualizzazione* negli ultimi anni ha guadagnato una sempre maggiore popolarità tra i professionisti e responsabili IT.
Presentandosi in primo luogo con la promessa di ridurre la complessità delle infrastrutture presenti all'interno dei moderni data center, le tecnologie di virtualizzazione sono penetrate in decine di aziende, società, ed organizzazioni.

Le tecniche ed i concetti alla base della virtualizzazione prendono le mosse già dagli anni '60 del secolo scorso, evolvendosi poi, sia nelle modalità di implementazione che negli obiettivi principali che si prefiggevano, fino ad arrivare, ormai da più di tre decenni, ad una forma di attuazione concettualmente simile a quella odierna. [2,3]
Inizialmente alla portata delle sole grandi imprese o delle grandi realtà scientifiche (laboratori di ricerca statali e/o militari), ricche di risorse e dotate di mezzi non certo comuni (erano i tempi dei grandi mainframe), **la sua tecnologia è oggi** disponibile per piattaforme e sistemi informatici **alla portata di realtà anche piccole** e, cosa non secondaria, **a costi molto più contenuti**, se non addirittura, in alcuni casi, gratuitamente (grazie ad iniziative *open-source*) o comunque inclusa nel prezzo dei prodotti che la implementano, quali ad esempio software di sistemi operativi.
L'argomento virtualizzazione è ormai divenuto così "caldo" che numerosissimi produttori software cercano di pubblicizzare i propri prodotti come appartenenti ed orientati a questo contesto tecnologico; in realtà possiamo dire, schematicamente ma in modo sostanzialmente corretto, che **sono riconducibili al contesto della virtualizzazione tutti quei prodotti e strumenti che concorrono**, in parte o integralmente, **ai seguenti obiettivi chiave**:

Obiettivi chiave Virtualizzazione

- Introdurre un ulteriore livello di astrazione tra le applicazioni e l'hardware;
- Favorire una riduzione dei costi e della complessità;

- Eliminare le ridondanze e massimizzare l'utilizzo delle infrastrutture IT;
- Migliorare i livelli e la qualità del servizio;
- Allineare in modo più efficace i processi IT agli obiettivi aziendali;
- Garantire l'isolamento delle risorse informatiche per una migliore affidabilità e sicurezza.

Cenni Storici

Prima di addentrarci nelle varie forme di virtualizzazione e per capire in modo più approfondito che cos'è e quali opportunità offre oggi, non si può prescindere da un **breve excursus sulla sua storia e le sue origini**.

La prima forma di virtualizzazione fu concepita negli anni '60 del secolo scorso e prese piede col nome di *time sharing* ("condivisione di tempo"). Christopher Strachey, primo professore di Calcolo all'Università di Oxford e leader del Programming Research Group, coniò questo termine nel suo scritto "*Time Sharing in Large Fast Computers*", usandolo quale estensione della "multiprogrammazione" ovvero l'esecuzione di più processi sullo stesso sistema contemporaneamente, sfruttando i momenti di inattività del processore.
In un contesto multiutente il time sharing avrebbe permesso a più programmatori di compilare ed eseguire contemporaneamente i loro programmi interagendo con il sistema centralizzato ciascuno dal proprio terminale. Il vantaggio era evidente visto che i primi computer mainframe erano estremamente costosi, e non sarebbe stato pensabile garantirne l'accesso esclusivo ad un singolo utilizzatore: con tale soluzione di gestione delle richieste multiutente da parte della CPU (detta *context switch*) si dava l'impressione ad ognuno di avere a disposizione il computer centrale interamente per sé.

La multiprogrammazione, insieme ad altre idee rivoluzionarie nate in quegli anni, cominciò a guidare l'innovazione, dando luogo in breve ad una serie di computer che irruppero con forza sulla scena. Due in particolare sono considerati parte integrante di quella sorta di percorso evolutivo che ha portato la virtualizzazione ad essere quella che conosciamo noi oggi, l'**Atlas** [4] e l'**IBM M44/44X** [5].

I primi progetti

Il primo di essi, progettato e gestito dal Dipartimento di Ingegneria Elettrica dell'Università di Manchester, il super-computer **Atlas**, mise a frutto i concetti di *time-sharing*, multiprogrammazione e controllo periferico condiviso per fornire ai suoi utilizzatori una velocità che nessun altro mainframe di quegli anni era in grado di dare. La gestione dei vari processi ***venne ripartita tra due componenti*** in base alla loro natura: uno si occupava dell'esecuzione dei programmi utente mentre l'altro, il *supervisor*, gestiva i processi del sistema operativo. Quest'ultimo gestiva anche le risorse chiave, come ad esempio il tempo di elaborazione del computer, avvalendosi anche di istruzioni speciali, dette *extracodes*, implementate appositamente per migliorarne la capacità di provvedere all'ambiente di calcolo per le istruzioni dei programmi utente. ***In sostanza, questa può essere considerata la nascita dell'hypervisor, il controllore delle macchine virtuali.***
Altri concetti che trovarono una prima forma di introduzione con l'Atlas furono quello di *memoria virtuale* (in origine detta *one-level store* per il fatto che, pur collocandosi su unità a livelli gerarchici distinti, tutta la memoria disponibile si presentava all'utente ad un unico livello di accessibilità), e quello delle tecniche di *paging* per la memoria di sistema, grazie alle quali il *supervisor* ed i processi del sistema operativo venivano eseguiti in un nucleo di memoria logicamente separato da quella usata dai programmi utente, sebbene le due parti fossero integrate nello stesso supporto fisico.
Si capisce quindi che, sotto molti aspetti, l'Atlas fu il primo passo verso la creazione di quello strato di astrazione che hanno in comune tutte le tecnologie di virtualizzazione.

La risposta di IBM, che non intendeva certo perdere il proprio ruolo di principale innovatore di computer, fu il Progetto **M44/44X**, sviluppato presso il Centro di Ricerca Thomas J. Watson di Yorktown, New York.
Tale progetto si fondava su un'architettura concettualmente simile a quella dell'Atlas -nella quale veniva tra l'altro usato per la prima volta il termine *virtual machines* (*macchine virtuali*, appunto)- e diventò il contributo principale di IBM agli emergenti sistemi di time sharing.

La macchina principale era un computer scientifico IBM 7044 (M44) con diverse macchine virtuali 7044, o 44X, simulate utilizzando sia hardware (la memoria virtuale) che software (la multiprogrammazione).
Negli anni successivi IBM perfezionò le tecnologie dei suoi sistemi con il Progetto dell'IBM 7094 nel quale introduceva il *Compatible Time Sharing System* [6], (**CTSS**), ovvero un sistema di time sharing compatibile con lo standard di elaborazione *batch* del sistema operativo utilizzato dalla macchina, il *Fortran Monitor System* (FMS), che consentiva di eseguire grossi carichi di lavoro tipicamente non interattivi (i *jobs*) in modo pianificato. Il CTSS non solo poteva eseguire una copia di FMS nel computer principale 7094, utilizzato come strumento primario per il flusso dello standard batch, ma poteva anche avviarne una copia in ogni macchina virtuale collegata ad esso. In questo modo i *jobs* in *background* avrebbero potuto accedere a tutte le periferiche quali nastri, stampanti, lettori di schede perforate e display grafici al pari dei jobs FMS in *foreground*, purché non interferissero con l'esecuzione di quest'ultimi o con qualsiasi altra risorsa collegata ad essi.

Alla fine degli anni sessanta un gruppo di ricercatori IBM del Cambridge Scientific Center (CSC), guidati da Norm Rassmussen e Robert Creasy, svilupparono con successo il primo sistema operativo per macchina virtuale basato su hardware completamente virtualizzato, il CP-40, precursore del popolare VM/370; quest'ultimo, rilasciato nel 1972, grazie al componente *Virtual Machine Monitor* (VMM, il controllore della macchina virtuale) avviato su hardware reale era in grado di eseguire numerose macchine virtuali indipendenti su copie virtuali di hardware. Ogni macchina virtuale del **VM/370** era in grado di eseguire un'unica installazione del sistema operativo IBM in modo stabile, indipendente e con elevate prestazioni [7].

Anche se i risultati di IBM sono stati i più influenti per la storia della virtualizzazione, non si può dire che questa fosse l'unica ad impegnarsi, già sul finire degli anni '60, per progetti basati sull'implementazione del *Time-Sharing*,

tanto che nella storia dei mainframe dell'epoca non possono essere taciuti riferimenti al *Los Alamos Scientific Laboratory* ed al *Lawrence Livermore Laboratory* con il loro **Cray Time-Sharing System** [8], un sistema operativo sviluppato nei primi anni '70 per i supercomputer Cray X-MP, utilizzati ampiamente in quegli anni anche per la ricerca nucleare dal Dipartimento di Energia degli Stati Uniti.

Ad ogni modo, proprio **attraverso i vari progetti sopra accennati, *il time-sharing* divenne presto ampiamente accettato e riconosciuto come un modo efficace per rendere più accessibili i primi *mainframe***.

Il paradigma accentrato degli albori dell'IT

Fin qui abbiamo ripercorso (sia pure in modo necessariamente sommario vista la vastità dell'argomento) i primi passi delle tecniche che hanno portato alla virtualizzazione, ed i primi progetti che ne avrebbero permesso l'evoluzione della tecnologia ad essa correlata fino ai giorni nostri.
Come visto gli sforzi maggiori erano all'inizio volti al perfezionamento della virtualizzazione dei server, ed **in questo primo tipo di contesto la virtualizzazione aveva lo scopo principale di fornire a più utilizzatori contemporanei l'esperienza di utilizzo esclusivo di grossi e costosi mainframe**; in sostanza, inizialmente, la virtualizzazione (e ancor prima il concetto di time-sharing) furono sostanzialmente un modo di aggirare l'enorme costo dei sistemi di elaborazione del tempo, ovvero un modo per sfruttare il più possibile (ovvero rendere fruibili a più persone possibili) sistemi intrinsecamente “rari” se non altro per il loro costo.

Questo stato di cose, ovvero la **predominanza totale di un paradigma centralizzato** basato su centri di elaborazione dati dotati di grandi e costosi computer centrali, rimase **sostanzialmente inalterato per buona parte degli anni '80** del secolo scorso; tale situazione si ritrovava anche nell'ambito delle telecomunicazioni ed anche se internet con il passare degli anni si stava diffondendo, per il momento era ancora lontana dal diventare quella rete di

interconnessione capillare che conosciamo oggi, assomigliando ancora, assai di più, ad una rete di collegamento dei grandi data center delle Università o delle grandissime realtà aziendali.

La proliferazione dei Server

Dagli anni '90 e fino ai giorni nostri, però, un insieme di fattori hanno cambiato il modo di concepire i data center, cambiando radicalmente i paradigmi organizzativi e architetturali degli stessi.

Si è passati così **da un modello centralizzato** fruibile soltanto ad un ristretto numero di grandi organizzazioni **ad uno decentralizzato** accessibile anche alla piccola e media impresa, ridimensionando le infrastrutture in una struttura orizzontale.

Il motivo scatenante che ha portato a questa mutazione del settore ICT si può identificare con la crescita su scala mondiale di Internet, grazie ad un miglioramento delle tecnologie disponibili e ad un abbassamento dei loro costi. L'uso di questo nuovo mezzo di informazione si è ingrandito a tal punto da dar vita ad una reazione a catena; da una parte le aziende e i privati, fino ad allora estranei al mondo telematico, hanno iniziato a trovare in internet un nuovo spazio dove poter offrire i propri servizi o dove pubblicizzarsi. Dall'altro si è assistito alla nascita di un grandissimo numero di piccole e medie imprese, in grado di rispondere a questo nuovo tipo di clientela. È la nascita della New Economy, che ha avuto grande successo in poco tempo perché in grado di offrire la possibilità di operare in un mercato globale abbattendo i costi di gestione e di non vincolare la propria clientela ad uno spazio definito quale può essere la sede fisica di una società o di un esercizio commerciale.

Ed i pilastri della New Economy furono le imprese in grado di offrire la creazione e la gestione di siti internet ai prezzi più accessibili.

La richiesta era giunta ad un punto tale che uno **sviluppo orizzontale** era diventato inevitabile. Ci si trovò di fronte ad una ***"proliferazione dei server"*;** invece dell'acquisto e del mantenimento di un unico host fisico centralizzato e delle relative periferiche necessarie per ogni applicazione, si giunse alla conclusione che ad ognuna di tali applicazioni era possibile dare il proprio

ambiente operativo, con hardware completo di I/O, potenza di elaborazione e memoria, lasciando spesso in condivisione soltanto le unità di storage ed alcune periferiche comuni. Così, mentre la quantità di applicazioni e ambienti applicativi dispiegati aumentava, anche il numero di server implementati all'interno dei data center cresceva a ritmi esponenziali. ***I server centralizzati erano visti come troppo costosi da acquistare e mantenere*** per le molte aziende non ancora fondate su tali piattaforme informatiche e, mentre i big frame, server dagli enormi carichi di lavoro delle grandi imprese continuavano a sopravvivere, il mercato dei server di fascia media e bassa andava incontro ad uno sviluppo sempre maggiore.

A colpo d'occhio la decentralizzazione offriva due benefici principali: la sicurezza e la stabilità, due fattori interconnessi allo stesso meccanismo di fondo. La ***stabilità*** proveniva dal mantenimento globale dell'infrastruttura in modo semplice ed efficace, grazie alle ***patch e agli aggiornamenti periodici facilmente applicabili senza interferire con gli altri sistemi in esecuzione***. Per lo stesso motivo, la decentralizzazione favoriva ***la sicurezza poiché un sistema compromesso si trovava fisicamente isolato dagli altri sistemi*** della rete.

In aggiunta, con la popolarità di Windows e delle piattaforme distribuite con i più leggeri sistemi *open source*, la promessa che molti speravano di ottenere includeva un migliore ritorno di investimento sulle attività e un minore *costo totale di proprietà* (TCO). La mercificazione delle piattaforme hardware e software a buon mercato ha dato ulteriore fondamentale impulso allo sviluppo di quel paradigma.

Il ritorno alla centralizzazione reinterpretata alla luce della Virtualizzazione

Le imprese si resero tuttavia ben presto conto che il ridimensionamento orizzontale necessario per nutrire le nuove istanze dei server, e la conseguente necessità di spazio e componenti per supportare tale espansione contrastava con gli obiettivi che li avevano mossi inizialmente. La proliferazione dei server si era intensificata principalmente con la domanda di spazio web, ma non solo; man mano che i processi IT diventavano più raffinati e pianificati allineandosi ai meccanismi di controllo e gestione delle grandi aziende, il *ciclo di vita dello sviluppo software* (SDLC) cominciava ad imporre una struttura rigida per lo

sviluppo di un prodotto software, definendone non solo le fasi (come la raccolta dei requisiti, l'architettura software e il design, i test, l'implementazione e la manutenzione), ma le regole stesse che guidano il processo di sviluppo attraverso ogni fase. In molti casi, le fasi risultavano sovrapposte, imponendo di avere la specifica configurazione *n-tier* dedicata. Più iterazioni della stessa applicazione erano necessarie per applicare il modello SDLC (*Software Development Life Cycle*) per lo sviluppo e garantire la qualità del prodotto, la fattibilità delle prove di carico, e il processo di produzione finale. Anche per queste necessità, la proliferazione dei server si è intensificata.
Il costo globale di sviluppo e produzione delle applicazioni è giunto ad un **maggior consumo** di potenza, un **minor spazio fisico disponibile** e un maggiore sforzo di gestione il quale, unito agli altri fattori, conta decine (se non centinaia) di migliaia di dollari in **costi di manutenzione** annuali per macchina. In aggiunta a questa gestione e manutenzione globale, la decentralizzazione ha **diminuito l'efficienza delle macchine**, lasciando la media di inattività del server ad una percentuale dell' 85-90 per cento. Queste inefficienze hanno eroso ulteriormente qualunque potenziale risparmio sul costo del lavoro promesso dalla decentralizzazione.

Così, contro ogni previsione, **negli ultimi anni si sta ritornando al paradigma centralizzato**. Proprio **la *Virtualizzazione***, però, **fornisce adesso una nuova interpretazione di questo paradigma originario, in quanto riduce i problemi sopra esposti scaturenti dalla proliferazione dell'hardware, ma riesce contemporaneamente a centrare gli obiettivi che avevano spinto in tale direzione**.

In questo nuovo contesto anche la virtualizzazione ha cambiato la propria declinazione, superando i confini originari del time sharing che la relegavano a mero stratagemma per l'utilizzo contemporaneo da parte di più operatori o processi di un'unica macchina costosa, ed arricchendosi di tutti i significati cui si accennava nell'introduzione e che ne fanno oggi uno dei principali strumenti per la semplificazione, la condivisione ed il consolidamento dei data center.

La situazione è molto diversa rispetto a trent'anni fa, sono diversi gli ambiti di utilizzo, i costi, le tecnologie ed anche i motivi che hanno spinto le aziende a questo ritorno; ma guardando la situazione dall'esterno ci si rende conto che la decentralizzazione è un capitolo che a poco a poco si sta chiudendo.

Considerazioni Finali

In conclusione del nostro breve excursus possiamo definire la ***virtualizzazione*** come una ***struttura informatica in cui si pratica un metodo di divisione delle risorse hardware*** in ambienti multipli di esecuzione, applicando uno o più concetti o tecnologie quali il ***partizionamento hardware* e *software***, il ***time sharing***, la ***simulazione*** parziale o totale della macchina, l'***emulazione***, la ***qualità del servizio***, e molti altri.

Proprio come si fece durante la fine degli anni '60 e gli inizi degli anni '70 con i primi VM/370 di IBM, la moderna virtualizzazione consente a più istanze del sistema operativo di essere eseguite simultaneamente su un singolo computer, anche se in modo molto più economico rispetto a quel periodo. Ognuna di tali istanze condivide le risorse disponibili sull'hardware fisico comune, come illustrato in *Fig.1.1*. Un programma, denominato ***Virtual Machine Monitor*** (***VMM***), o ***Hypervisor***, controlla l'uso e l'accesso alla CPU, alla memoria, alle unità di storage e alle risorse di rete presenti al livello sottostante.
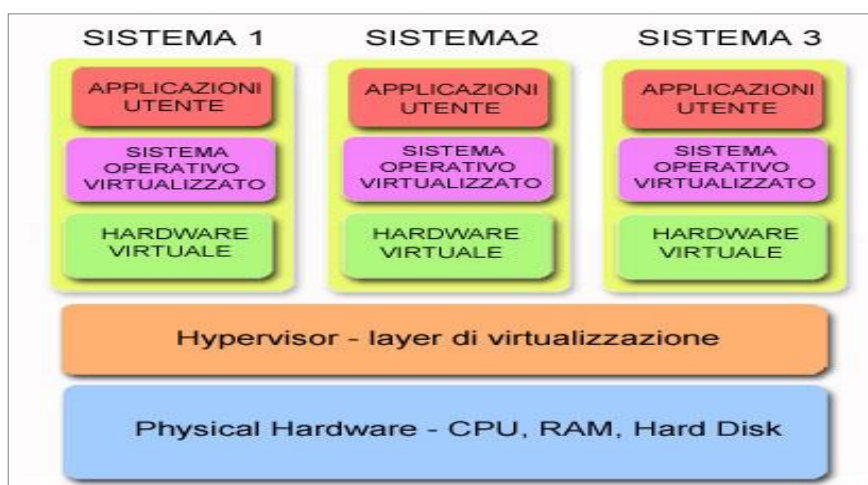
Fig. 1.1 – Le macchine virtuali girano sull'hardware fisico grazie al VMM

4 - I “PERCHÈ” ED I “COME” DELLA VIRTUALIZZAZIONE

Meccanismi e Benefici Principali

La virtualizzazione ricopre un ruolo fondamentale nell'ottimizzazione dei sistemi. A prima vista potrebbe semplicemente apparire come un modo per ridurre e semplificare l'infrastruttura dei server, ma invece può rivelarsi uno strumento per trasformare il modo di concepire il data center nel suo complesso. In *Fig.4.1* si illustra il modello di ottimizzazione dei sistemi. Il consolidamento fisico è la base necessaria per poter affrontare i passi successivi, che sono il consolidamento logico e la razionalizzazione complessiva dei sistemi e delle applicazioni attraverso l'identificazione di applicazioni che sono inutili o ridondanti e possono quindi essere eliminate.
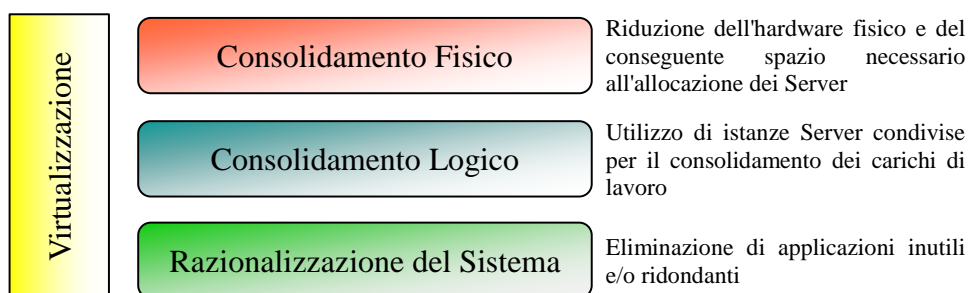
Fig. 4.1 – Ruoli della virtualizzazione nell'ottimizzazione dei server

I benefici che giustificano lo spostamento di un’organizzazione IT verso un’infrastruttura virtuale si possono semplificare in tre concetti principali:

- **Consolidamento**. Aumenta lo sfruttamento dei server, semplifica la migrazione dei *software legacy* (programmi tipicamente forniti da terze parti, sviluppati anni addietro con le tecnologie disponibili all'epoca e che, sebbene abbiano subito interventi di manutenzione, non sono mai stati rimpiazzati), ospita sistemi operativi misti e semplifica gli ambienti di sviluppo e di test.
- **Affidabilità.** Isola eventuali difetti e/o guasti causati dal software, all'occorrenza rialloca le VM (Macchine Virtuali) esistenti o ne crea nuove di *failover*, permettendo di trasferire funzioni di sistema svolte da una macchina virtuale compromessa ad una nuova perfettamente funzionante.

- **Sicurezza.** Contiene gli attacchi informatici attraverso l'isolamento delle minacce e dei bug, permettendo anche di applicare differenti impostazioni di sicurezza ad ogni VM coesistente sullo stesso host.

Consolidamento

L'obiettivo alla base delle tecniche di **consolidamento** si può riassumere con i termini *aggregare* e *unificare*.

Nel caso della virtualizzazione, i carichi di lavoro vengono concentrati in un minor numero di piattaforme fisiche in grado comunque di sostenere le loro richieste di risorse quali CPU, memoria e I/O.

Molto spesso nei data center i carichi di lavoro in esecuzione sono ben lontani dallo sfruttare pienamente l'hardware su cui girano, con conseguente spreco delle infrastrutture e abbassamento dei profitti. Attraverso il consolidamento, la virtualizzazione permette di concentrare in modo strategico le istanze dei server o dei sistemi operativi con i rispettivi carichi di lavoro ponendoli su hardware condiviso con una disponibilità di risorse sufficienti a soddisfarne la domanda.

In passato si pensava che i server non dovessero essere fatti lavorare in prossimità del loro limite di capacità; è invece vero il contrario. Al fine di massimizzare l'investimento, i server devono lavorare il più possibile vicino al loro carico massimo, ovviamente senza saturare completamente le risorse disponibili compromettendo le prestazioni dei carichi di lavoro o dei processi in esecuzione. Seguendo una corretta pianificazione e comprendendo quanto incidono tali processi, **la virtualizzazione contribuisce all'incremento del livello di utilizzo dei server riducendo nel contempo il numero di piattaforme fisiche necessarie**.

Un discorso a parte può poi essere fatto per i cosiddetti *sistemi legacy*; con tale appellativo (che potrebbe essere tradotto un po' letteralmente come “*sistemi ereditati”*) ci si riferisce tipicamente a sistemi esistenti nella realtà (azienda o organizzazione) considerata che, pur essendo tipicamente obsoleti come tecnologia, continuano ad essere utilizzati per ragioni di convenienza economica (costi delle licenze dei prodotti aggiornati), di convenienza pratica (perché talvolta soddisfano le necessità presenti meglio dei corrispondenti prodotti di nuova

concezione) ovvero per la materiale impossibilità di ottenere versioni più nuove (perché ad esempio forniti da terze parti che hanno abbandonato lo sviluppo del software in questione).
Quale che sia in ogni caso il motivo dell'obsolescenza tecnologica dei sistemi legacy, è chiaro come questi, utilizzando tecnologie meno recenti, siano spesso difficili da utilizzare nel contesto di piattaforme (hardware e di S.O.) di ultima generazione; ad esempio l'hardware dei server si è sviluppato a livelli tali che spesso è diventato incompatibile con i sistemi operativi e le applicazioni legacy. L'utilizzo di tecnologie e di processori più recenti, dei relativi chipset supportati e dei bus ad alta velocità può, il più delle volte, penalizzare i sistemi legacy, se non addirittura renderli inutilizzabili.
Proprio per questo motivo, in passato, sono stati mantenuti -man mano che i data center migravano verso nuove tecnologie e generazioni di apparati- anche vecchi server utilizzati esclusivamente per far girare ambienti -spesso superati- compatibili con le richieste dei sistemi legacy.
Anche su questo fronte **la virtualizzazione fornisce una soluzione di consolidamento dei sistemi *legacy* verso sistemi di nuova generazione**, consentendo la migrazione di questi sistemi in modo semplice, fornendo piattaforme (VM) idonee e retrocompatibili con l'esecuzione delle loro istanze.

Un altro aspetto del consolidamento che discende dalla virtualizzazione, è quello di **ricondurre anche sistemi operativi diversi ad una medesima piattaforma hardware**. In passato i sistemi operativi erano spesso legati ad una specifica piattaforma hardware. Questo poneva seri limiti ad aziende ed organizzazioni, costringendole spesso a fare grandi investimenti in hardware al fine di mantenere le loro applicazioni di business più critiche.
Attualmente però molti dei più comuni sistemi operativi disponibili possono essere eseguiti su una vasta gamma di architetture server, la più popolare delle quali è l'architettura x86. In essa è possibile eseguire Windows, UNIX o una delle distribuzioni Linux a propria scelta.
Pertanto, le tecnologie di virtualizzazione costruite sopra all'architettura x86

possono ospitare ambienti eterogenei. Sistemi operativi multipli, compresi quelli menzionati precedentemente, possono essere consolidati sullo stesso hardware fisico, riducendo ulteriormente i costi di acquisizione e di manutenzione.

Il consolidamento, permette anche, tipicamente, di snellire gli ambienti di sviluppo e di test generalmente necessari in ambito aziendale prima di mettere *in produzione* una nuova versione di software o, peggio, un nuovo tipo di applicativo destinato a rimpiazzare uno precedentemente in uso.

Invece di avere una proliferazione incontrollata dell'infrastruttura, dovuta alla nascita di nuovi progetti o nuovi rilasci o al mantenimento di applicazioni esistenti, la virtualizzazione consente di consolidare molti di quei carichi di lavoro su un minor numero di server fisici.

Lo strato di astrazione introdotto dalla **virtualizzazione**, infine, pur facendo girare ogni VM come se fosse in esecuzione su hardware dedicato, **consente tuttavia una flessibilità che non si avrebbe in un contesto puramente fisico**. Le partizioni possono infatti all'occorrenza essere riassegnate per servire altre funzioni.

Si immagini un server che ospiti un'applicazione client-server utilizzata solo durante l'orario che va dalle 9:00 alle 18:00 dal lunedì al venerdì, un altro server che abbia in esecuzione processi batch atti ad eseguire determinati controlli sul lavoro della giornata chiudendo le operazioni ogni notte, e un altro ancora responsabile dei lavori di manutenzione sui dati nel week-end. In un mondo puramente fisico, essi esisterebbero come tre server dedicati, molto utilizzati durante le ore di rispettivo esercizio, ma inattivi quando non svolgono la loro attività. Questo provocherebbe un grande sottoutilizzo e di conseguenza renderebbe più caro l'investimento. La virtualizzazione risolve il problema consentendo ad una singola partizione logica o fisica di essere riassegnata all'occorrenza a ciascuna funzione. Nei giorni feriali ospiterebbe l'applicazione client / server di giorno ed eseguirebbe i processi batch durante la notte. Durante i fine settimana, sarebbe riassegnata alle attività di manutenzione dei dati, per ritornare poi ad ospitare l'applicazione client / server la mattina del lunedì.

Questa flessibilità consente alle organizzazioni IT di utilizzare le partizioni “part-

time”, eseguendo i processi business principali nello stesso modo in cui lo farebbero i server fisici, ma realizzando anche una riduzione dei costi pur mantenendo elevati livelli di affidabilità.

Affidabilità

L'**affidabilità** è diventata sempre più un punto cardine per le organizzazioni IT. Essa ha una relazione diretta con la **disponibilità del sistema**, i tempi di *uptime* delle applicazioni e, di conseguenza, con i servizi erogati e quindi, direttamente o indirettamente, con i redditi generati.

Società ed Organizzazioni sono spesso disposte a investire pesantemente nelle proprie infrastrutture server per assicurare che almeno le loro applicazioni più critiche restino costantemente online e che, quindi, il loro funzionamento non sia mai interrotto.

Investendo in hardware e software aggiuntivi per monitorare errori e guasti del software, è certamente possibile gestire imprevisti e tempi di inattività non programmati riducendo al minimo, se non addirittura evitando, qualsiasi interruzione. Così facendo, però, i costi aumentano a dismisura.

Le tecnologie di virtualizzazione, invece, sono già intrinsecamente predisposte ad affrontare questa situazione, fornendo un forte isolamento tra le macchine virtuali in esecuzione. Un errore di sistema in una macchina virtuale o in una partizione non pregiudicherà l'esecuzione delle altre partizioni in esecuzione sulla stessa piattaforma hardware. Per contro **un eventuale guasto all'hardware sul quale fisicamente si appoggia l'esecuzione di una macchina virtuale, provocherà** -a prescindere dalla sua gravità**- un fermo limitato al tempo strettamente necessario a far ripartire la VM stessa su un hardware diverso**.

Questo isolamento logico protegge le macchine virtuali a livello più basso rendendole inconsapevoli, e quindi non influenzabili, da errori o guasti al di fuori delle loro assegnazioni.

Un altro elemento che, in un contesto puramente fisico, fa lievitare i costi è **l'implementazione ed il mantenimento in funzione di server di standby o di fault tolerance** necessari anche per mantenere il sistema disponibile durante i periodi di interruzioni pianificate e non. Spesso tali server sono sostanzialmente

identici (come dotazione hardware) a quelli di produzione che devono sostituire in caso di guasti e/o fermi tecnici, ma -salvo durante queste rare interruzioni- generalmente inattivi. Si riducono così ad essere sotto-utilizzati, fornendo poco valore all'azienda sebbene il loro costo sia molto elevato.

La virtualizzazione aiuta a risolvere anche questo problema consentendo all'occorrenza la dotazione di **partizioni aggiuntive just-in-time** (JIT, letteralmente *sul momento*, ovvero assegnate in modo dinamico nel momento del bisogno) **o on-demand** (su richiesta). Ad esempio, è possibile predisporre **VM "di scorta"** costruite e configurate con il medesimo sistema operativo e le stesse applicazioni delle corrispondenti VM di produzione, che possono essere **tenute in uno stato di inattività (spegnendole o mettendole in sospensione), pronte per essere attivate quando si verifica un errore.**

In questo scenario è possibile pensare ad un numero di VM "di scorta" decisamente superiore al numero di server hardware di emergenza: al limite si potrebbe anche pensare di avere in azienda un unico hardware dedicato al fault tolerance sul quale viene eseguita "al volo" una o più VM di scorta in corrispondenza di guasti o fermi non previsti di una o più VM dell'ambiente di produzione.

Si capisce che in questo modo si riducono drasticamente i costi e soprattutto il sottoutilizzo dell'hardware acquistato (si potrebbe anche pensare che il server dedicato al fault tolerance, esegua VM dedicate a processi meno critici -e pertanto immediatamente interrompibili per far posto a quelli delle VM "di scorta" in caso di bisogno- nei momenti in cui l'ambiente di produzione funziona regolarmente).

Sicurezza

La stessa tecnologia che fornisce isolamento agli errori delle applicazioni è anche in grado di fornire **isolamento alle falle di sicurezza**. Se una particolare partizione (VM) venisse compromessa, essendo isolata, circoscriverebbe il problema, evitando di compromettere anche le altre VM.

Soluzioni a questo eventuale problema possono essere attuate anche isolando ulteriormente le partizioni compromesse e le istanze del sistema operativo negando loro le risorse necessarie affinché esse esistano. I cicli della CPU

possono essere ridotti, l'accesso alla rete e ai sistemi di I/O possono essere interrotti, fino a raggiungere, nei casi più estremi, il blocco totale del sistema.
Sarebbe difficile, se non impossibile, eseguire questi compiti se l'istanza compromessa fosse in esecuzione direttamente su un host fisico.
Quando si stanno consolidando i carichi di lavoro **attraverso la virtualizzazione, le configurazioni di sicurezza possono essere specificate per ogni singola VM e non per tutto il server nel suo insieme.** Un esempio di ciò potrebbe essere l'utente Administrator dei S.O. Windows, ovvero l'account super-user dei sistemi Linux e Unix.

Le applicazioni, consolidate in un unico sistema operativo, se eseguite direttamente su di un server fisico condividerebbero varie impostazioni di sicurezza; in particolare, l'accesso root sarebbe lo stesso per tutte.
Al contrario, quando gli stessi carichi di lavoro vengono consolidati su partizioni virtuali, ciascuna partizione può essere configurata con credenziali diverse, mantenendo così l'isolamento dell'accesso al sistema con privilegi amministrativi.

Quindi, anche sotto l'aspetto della sicurezza (oltre che per quelli sopra trattati del consolidamento e dell'affidabilità), la virtualizzazione si delinea come la scelta migliore in quasi tutte le realtà, piccole o grandi che siano. Essa risponde a pieni voti alle esigenze di contenimento dei costi e alla diminuzione di spazio all'interno dei data center, attuando i vari mandati delle varie realtà considerate in tempi più stretti e con ottimi risultati.

Virtualizzazione basata sul Software

Benché ci siano vari modi per virtualizzare le risorse informatiche utilizzando un VMM (Virtual Machine Monitor), ognuno di essi punta allo stesso obiettivo: consentire ai sistemi operativi di venire eseguiti in modo indipendente e isolato l'uno dall'altro, ed esattamente come se venissero avviati direttamente su una piattaforma hardware.
Sebbene le tecnologie di *virtualizzazione hardware-based*, che virtualizzano e astraggono l'hardware completamente (riprendendo e sviluppando concetti e

soluzioni di cui si è accennato parlando dei primi sistemi Atlas e VM/370) esistano ancora, esse tendono però ad essere costose e poco flessibili.

Di conseguenza sono sorti moltissimi software Hypervisor e VMM per eseguire la ***virtualizzazione*** attraverso meccanismi ***software-based***. Essi garantiscono un grado di isolamento in cui il basso livello, nucleo centrale dell'architettura della CPU, è portato vicino ai livelli software dell'architettura per consentire ad ogni macchina virtuale di avere il proprio ambiente dedicato.

Lo stretto rapporto tra l'architettura della CPU e i sistemi operativi virtualizzati è alla base del funzionamento dei moderni sistemi di virtualizzazione, pertanto un breve approfondimento di tale relazione è fondamentale per capire i meccanismi di funzionamento della virtualizzazione stessa.

Relazione fra CPU, S.O. e VMM

Le architetture hardware ideali sono quelle in cui il Sistema Operativo e la CPU sono progettati e costruiti l'uno per l'altra, e risultano quindi associati strettamente. Un corretto uso delle chiamate di sistema richiede un attento coordinamento tra il sistema operativo e la CPU.

Questa sorta di simbiosi fra CPU e S.O. si traduce in numerosi vantaggi in materia di sicurezza e stabilità di tutta l'architettura.

Un esempio storico di tale approccio "simbiotico", è stato il MULTICS (acronimo di Multiplexed Information and Computing Service), uno dei primi sistemi operativi fortemente orientato al time-sharing sviluppato a partire dal 1964 e progettato per una particolare architettura della CPU, (progettata in tandem dal MIT e dalla General Electric), a sua volta pensata appositamente per esso.

Ciò che a suo tempo rese il MULTICS speciale fu il suo approccio volto ad isolare il più possibile le varie operazioni software per eliminare il rischio che un componente eventualmente divenuto instabile potesse compromettere o destabilizzare anche altri componenti. Per far questo il MULTICS introdusse i cosiddetti "protection rings", (anelli di protezione), delle tecniche formali volte a separare il sistema operativo, "trusted", dai programmi utente, "untrusted".

MULTICS fu un progetto assai elaborato che, prevedendo ben otto di questi anelli

di protezione, riuscì nell'intento di isolare i vari livelli astraendo dall'interazione con l'hardware; tuttavia l'approccio progettuale impiegato per il MULTICS si dimostrò troppo costoso per potersi affermare a livello globale come paradigma delle architetture dei moderni computer.

La Famiglia x86

*L'**architettura** CPU più comune utilizzata nei moderni computer è l'IA-32, o **x86-compatibile.***

A partire dal chipset 80286 (introdotto nel febbraio del 1982), la famiglia x86 fornì due metodi principali di indirizzamento della memoria: la *modalità reale* (limitata ad 1MB e senza supporto al multitasking) e la *modalità protetta* (gestita direttamente in hardware da una MMU -Memory Manager Unit- che si occupa di impedire che un processo possa corrompere una porzione di memoria in uso da un altro processo contemporaneamente in esecuzione sul medesimo computer).

Nel chipset 80386 e successivi venne introdotta una terza modalità chiamata *modalità virtuale 8086*, o VM86, che permetteva l'esecuzione di programmi scritti per la modalità reale, ma eludendone le limitazioni senza dover passare alla modalità protetta.

La modalità reale, limitata ad un singolo megabyte di memoria, diventò rapidamente superata, ed anche la modalità virtuale, limitata ad operazioni a 16-bit, divenne obsoleta quando i sistemi operativi a 32 bit iniziarono ad essere ampiamente disponibili per l'architettura x86.

La modalità protetta, componente fondamentale dell'x86, venne invece dotata di numerose nuove funzionalità per supportare il multitasking, e “salvò” di fatto l'architettura x86. Le nuove funzionalità introdotte comprendevano ad esempio la segmentazione dei processi e il supporto hardware per la memoria virtuale e la commutazione dei processi.

Nella famiglia x86 la modalità protetta utilizza quattro livelli privilegiati, o *Ring*, numerati da 0 a 3. La memoria di sistema è divisa in segmenti e ogni segmento viene assegnato e dedicato ad un particolare Ring. Il processore utilizza il livello privilegiato per determinare cosa può e non può essere fatto con il codice o i dati contenuti nel segmento. Il termine *“Ring”* deriva proprio dal sistema MULTICS,

dove i livelli privilegiati erano visualizzati appunto come una serie di *anelli* concentrici. Il Ring-0 è considerato l'anello più interno, il *kernel*, al quale è assegnato il controllo totale del processore. Il Ring-1 e 2 attuano l'esecuzione dei servizi del sistema operativo e dei driver delle periferiche. Infine il Ring-3, l'anello più esterno, legato all'esecuzione delle applicazioni, fornisce solo accesso limitato, come illustrato nella *Fig.4.2*.
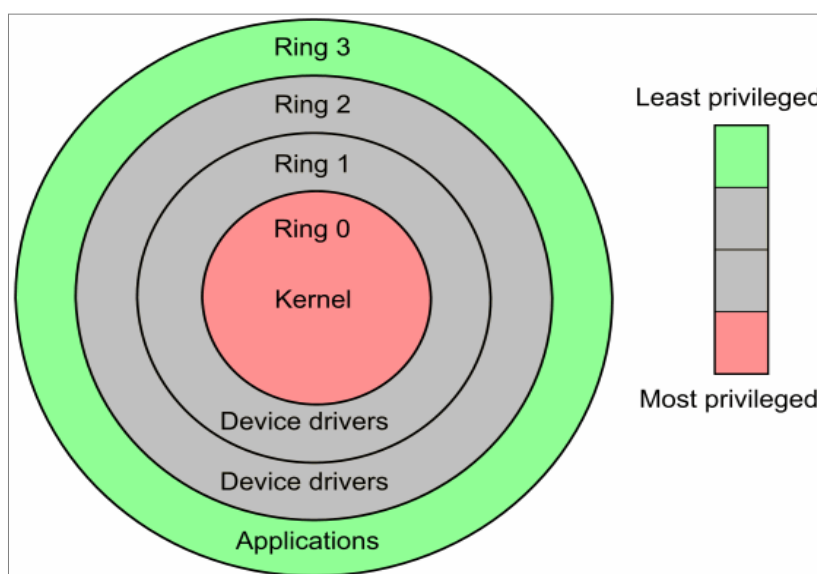
Fig. 4.2 – Anelli privilegiati nell'Architettura x86

Nella moderna architettura dei sistemi operativi Windows, Linux e la maggior parte dei sistemi UNIX viene ancora utilizzata la struttura ad anelli, anche se è stata ridotta ad un approccio a doppio strato che utilizza solo gli anelli 0 e 3, come schematizzato in *Fig.4.3*:
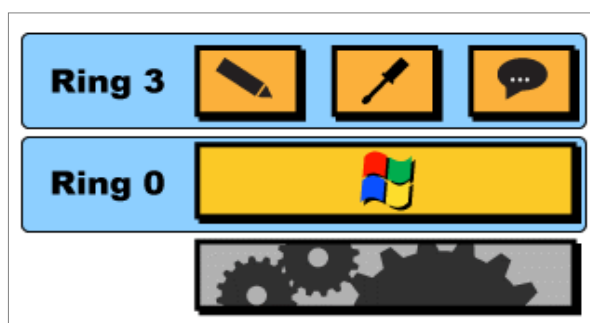
Fig. 4.3 – Anelli privilegiati nei moderni S.O.

Al Ring-0, comunemente chiamato Supervisor Mode (Modalità Supervisore), è stato demandato il compito assegnato in precedenza ai tre anelli interni: Ring-0,1 e 2, mentre il Ring-3, noto come User Mode (Modalità Utente), ha mantenuto le stesse funzioni.

Meccanismi di sicurezza implementati direttamente in hardware impongono restrizioni al Ring-3 limitando l'accesso di codice ai segmenti, al paging, e alle operazioni di input / output. Se ad esempio un programma utente eseguito sul Ring-3 cerca di indirizzare della memoria al di fuori dei segmenti di sua competenza, un hardware interrupt (interruzione hardware) blocca l'esecuzione di codice.

Tipi di VMM

Come si è visto il "Supervisor Mode" è la modalità, su un processore x86, che consente l'esecuzione di tutte le istruzioni, comprese le istruzioni privilegiate come le operazioni di I / O e di gestione della memoria.

Il sistema operativo funziona normalmente in tale modalità (Ring-0). Siccome il Ring-3 è basato sul Ring-0, qualsiasi compromissione o instabilità del sistema si ripercuote direttamente sui processi in esecuzione nello User Mode del Ring-3.

Per isolare il Ring-0 "percepito" da ciascuna macchina virtualizzata (o *guest*), diviene allora necessario spostarlo ad un livello più esterno del Ring-0 "reale". In questo modo, un errore o guasto del Ring-0 percepito da un determinato guest virtualizzato non incide sul Ring-0 "reale", e conseguentemente sul Ring-3, di qualsiasi altro guest. I Ring-0 percepiti dai guest possono risiedere in uno dei Ring-1, 2 o 3 delle architetture x86. Naturalmente, quanto più la percezione del Ring-0 è lontana dal vero Ring-0, tanto più sarà difficile eseguire le operazioni hardware direttamente, con conseguente riduzione delle prestazioni.

La virtualizzazione muove quindi il Ring-0 nel modello ad anelli privilegiati mettendo il Virtual Machine Manager, o VMM, in uno degli anelli, che a sua volta rappresenta l'implementazione del Ring-0 sulle macchine virtuali guest. I sistemi operativi guest vengono eseguiti su questo particolare ***Ring-0 "percepito"***, mentre il VMM gestisce l'interazione reale con la sottostante piattaforma hardware per l'accesso alla CPU, alla memoria e alle risorse di I / O.

Esistono due tipi di VMM che indirizzano la rappresentazione del Ring-0:

- **VMM Tipo 1**: ***software che viene eseguito direttamente*** su di una piattaforma hardware ***sul*** vero ***Ring-0***. I sistemi operativi guest (cioè delle macchine virtuali) vengono eseguiti ad un livello al di sopra di quello hardware, consentendo un vero isolamento di ciascuna macchina virtuale.
- **VMM Tipo 2**: ***software che viene eseguito all'interno di un sistema operativo***, di solito ***sul Ring-3***. Per quanto visto su gli anelli di privilegio nell'architettura x86, si capisce che in questo caso *il Ring-0 percepito dal sistema operativo delle macchine virtuali è molto più lontano dall'effettiva piattaforma hardware*. La virtualizzazione effettuata con questo tipo di VMM tende ad avere ***performance inferiori*** a quella dei VMM di tipo 1, ma offre altri vantaggi, primo fra tutti quello della ***maggior compatibilità*** hardware (infatti i VMM di tipo 2 supportano di fatto tutto l'hardware supportato dal S.O. all'interno del quale sono eseguiti, ovvero il cosiddetto S.O. host).

Le differenze di approccio dei due tipi di VMM sono ben schematizzate dalla *Fig.4.4*.
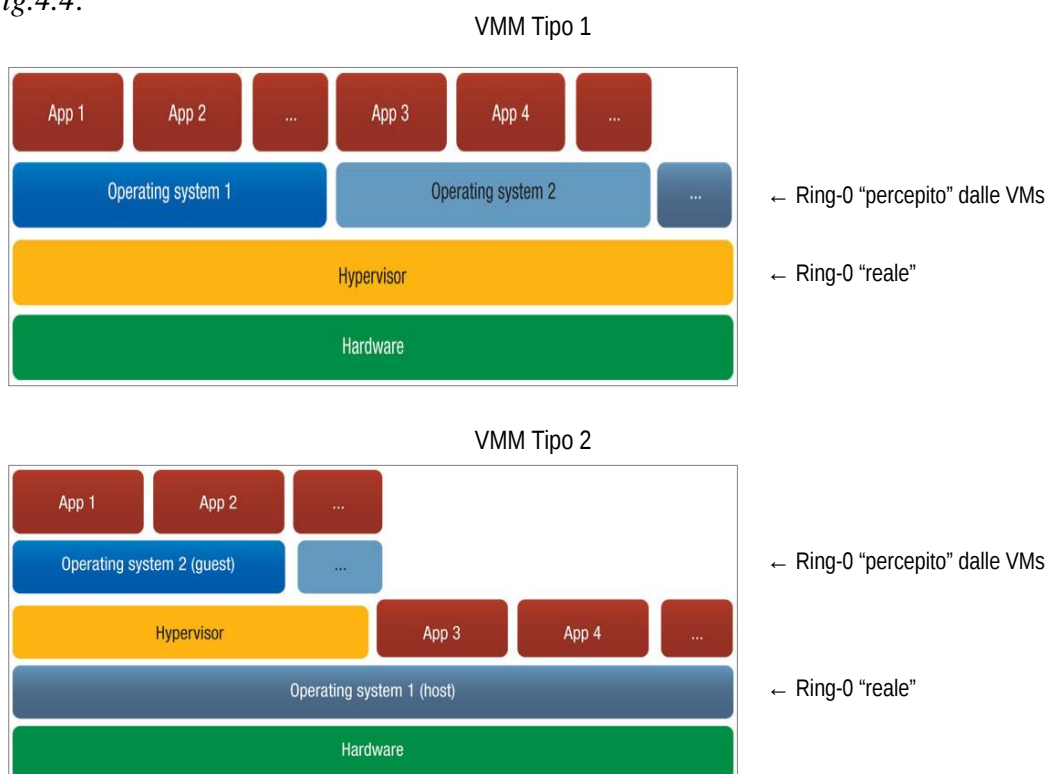
Fig. 4.4 – VMM Tipo1 e VMM Tipo2

Il ruolo del VMM

Per quanto detto, **per creare le partizioni virtuali in un server, viene eseguito un sottile strato di software chiamato VMM, *Virtual Machine Manager*** (o anche *Virtual Machine Monitor*), direttamente sulla piattaforma hardware fisica (VMM Tipo1) ovvero all'interno di un sistema operativo host (VMM tipo2).
In ognuno dei due casi, **al di sopra del VMM possono essere eseguiti una o più macchine virtuali** ciascuna con il proprio sistema operativo guest e le relative applicazioni.
Il VMM è quindi il centro della virtualizzazione dei server. Gestisce da una parte le risorse hardware e dall'altra le richieste dei sistemi operativi guest e delle loro applicazioni. Assegna ad ogni guest un insieme virtuale di CPU, memoria, I/O, e risorse su disco basato sull'hardware fisico disponibile o sulla base di una selezione personalizzata dell'hardware sottostante.
Volendo illustrare il ruolo del Virtual Machine Manager e le considerazioni che devono essere fatte in margine alla sua progettazione, non si può non citare i **requisiti di virtualizzazione di Popek e Goldberg**: generalmente indicati come la fonte di riferimento originale per i criteri che devono essere soddisfatti dal VMM, essi definiscono di fatto le condizioni delle architetture dei computer atte a sostenere la virtualizzazione.
Scritti nel 1974 per i computer di terza generazione di quel periodo [9], hanno generalizzato le **condizioni che il software che fornisce l'astrazione di una macchina virtuale, cioè il VMM, deve soddisfare**; tali condizioni sono:

- **Equivalenza.** Un programma in esecuzione attraverso il VMM deve dimostrare un comportamento prevedibile che è sostanzialmente identico a quello mostrato durante la sua esecuzione direttamente sulla piattaforma hardware sottostante. Questa condizione viene chiamata anche *"Fidelity"*.

- **Controllo delle risorse.** Il VMM deve avere il controllo completo in ogni momento delle effettive risorse hardware virtualizzate per i sistemi operativi guest. Questa condizione viene chiamata anche *"Safety"*.

- **Efficienza.** Un grande numero di istruzioni macchina devono essere eseguite senza l'intervento della VMM, ma dall'hardware stesso. Questa condizione viene chiamata anche *“Performance”*.

Secondo Popek e Goldberg, ***il problema che gli sviluppatori del Virtual Machine Manager devono affrontare è la creazione di un VMM che soddisfi le condizioni precedenti quando si opera all'interno delle caratteristiche*** dell'Istruction Set Architecture (ISA) ***della specifica piattaforma hardware utilizzata***.
Il set ISA può essere suddiviso in tre gruppi di istruzioni: quelle privilegiate, di controllo, e di funzionamento.
Le istruzioni privilegiate vengono bloccate se il processore è in User Mode mentre vengono eseguite se è in Supervisor Mode. Le istruzioni di controllo sensibile tentano di modificare la configurazione delle reali risorse della piattaforma hardware. Infine le istruzioni di funzionamento sono quelle il cui comportamento o il cui risultato dipende dalla configurazione delle risorse.
I VMM devono lavorare con ogni gruppo di istruzioni, mantenendo le condizioni di *equivalenza*, *controllo delle risorse* ed *efficienza*.
Tutti i VMM di oggi soddisfano almeno le prime due condizioni: equivalenza e controllo delle risorse. Per ottenere ciò gestiscono in modo efficace i sistemi operativi guest e la piattaforma hardware sottostante tramite l'***emulazione***, l'***isolamento***, l'***allocazione*** e l'***incapsulamento***; vediamo di seguito il significato di tali concetti:

- **Emulazione**. L'emulazione è importante per tutti i sistemi operativi guest. Il VMM deve presentare un ambiente hardware completo, (la cosiddetta *macchina virtuale* o, brevemente, VM), sia per i sistemi operativi che per le applicazioni. Il sistema operativo e le applicazioni sono all'oscuro del fatto che condividono le risorse hardware con altre applicazioni. *L'emulazione è la chiave per soddisfare la proprietà di equivalenza*.
- **Isolamento**. L'isolamento è importante per un ambiente sicuro e affidabile.

Attraverso l'astrazione hardware, *ogni macchina virtuale deve essere sufficientemente separata e indipendente dalle operazioni e dalle attività delle altre* macchine virtuali. I guasti che si verificano in una singola macchina virtuale non devono coinvolgere le altre; ciò fornisce sicurezza e disponibilità ad alti livelli.

- **Allocazione**. Il VMM deve *assegnare con metodo le risorse* della piattaforma *alle macchine virtuali* che gestisce. Risorse di elaborazione, memoria, rete e unità di storage devono essere bilanciati per ottimizzare le prestazioni e allineare i livelli di servizio con i requisiti di business. *Attraverso l'allocazione il VMM soddisfa la proprietà del controllo delle risorse e, in parte, anche la proprietà di efficienza*.

- **Incapsulamento**. L'incapsulamento, anche se non è specificato nei requisiti Popek e Goldberg, *permette ad ogni pila di software di essere altamente portabile*, in grado di essere copiata o spostata da una piattaforma che esegue il VMM ad un'altra. In alcuni casi è possibile "migrare a caldo" le macchine virtuali, cioè spostarle da un supporto ad un altro senza doverle spegnere, cioè mantenendole in esecuzione. L'incapsulamento deve contenere le informazioni di stato per mantenere l'integrità della macchina virtuale trasferita.

Prendendo in considerazione l'architettura più diffusa, la IA-32 (x86), tutto il software viene eseguito necessariamente in uno dei quattro Ring di privilegio.

Il sistema operativo di norma ***viene eseguito nel Ring-0***, che offre un accesso privilegiato alla più ampia gamma di risorse, del processore e della piattaforma. ***Le singole applicazioni solitamente vengono eseguite nel Ring-3***, che limita alcune funzioni (come la mappatura della memoria) che potrebbero disturbare altre applicazioni. In questo modo, il sistema operativo mantiene il controllo per garantire un funzionamento ottimale.

Poiché il VMM deve avere il controllo privilegiato delle risorse della piattaforma, la soluzione che appare più naturale è eseguire il VMM nel Ring-0 e i sistemi operativi guest nel Ring-1 o nel Ring-3. ***Tuttavia, i sistemi operativi moderni sono stati specificamente progettati per l'esecuzione sul Ring-0***, creando una

sorta di problema. In particolare, esistono alcune istruzioni privilegiate in grado di controllare le risorse critiche della piattaforma. Tali istruzioni vengono utilizzate in modo occasionale, ma quando un sistema operativo non viene eseguito nel Ring-0 una di queste istruzioni può creare un conflitto, provocando un errore di sistema o una risposta non corretta. La sfida incontrata dai VMM per l'architettura IA-32 (x86) è il mantenimento dei requisiti Popek e Goldberg mentre si lavora con l'ISA IA-32.

Tipi di Virtualizzazione

Si è già parlato del fatto che esistono molte forme di virtualizzazione nella moderna tecnologia dell'informazione, ma ad oggi la più comune e diffusa è ancora la virtualizzazione dei server, tipico caso generalmente associato a questo termine.
Le piattaforme e architetture CPU più diffuse su cui viene implementata la virtualizzazione dei server sono le IA-32 o x86. ***Le sfide poste dall'architettura ISA x86 e dai requisiti Popek e Goldberg hanno portato a diversi approcci per lo sviluppo dei VMM***. Benché ci siano diverse implementazioni della VMM per x86, si possono riassumere in ***quattro distinte categorie*** [10,11]: ***Full Virtualization***, ***Paravirtualizzation***, ***Operating System Virtualization*** e ***Native (o Hardware Assisted) Virtualization***. Vediamo di seguito, rapidamente, caratteristiche e principali vantaggi e svantaggi di ciascuna di esse.

A – Full Virtualization

È una tecnica di virtualizzazione che ***fornisce un ambiente in grado di simulare completamente l'hardware sottostante***. Il risultato è un sistema in cui tutto il software in grado di essere eseguito su hardware “reale” può essere eseguito senza modifiche anche su macchina virtuale.

VANTAGGI: questo tipo di virtualizzazione garantisce il più ampio supporto alla maggior parte dei sistemi operativi e applicazioni guest, che possono “girare” sulle macchine virtuali senza alcuna modifica (e tipicamente senza alcuna “consapevolezza” di avere a che fare con hardware virtuale). Le varie VM risultano completamente isolate l'una dall'altra e dal VMM stesso.

SVANTAGGI: nella sua forma teorica la full virtualization non è immediatamente realizzabile nell'architettura x86 in quanto le *istruzioni privilegiate* dei *S.O. guest* non sono direttamente eseguibili nel *ring-0 percepito* (perché questo è necessariamente *al di sopra del ring-0 reale* anche nel caso di VMM di Tipo-1); per questo motivo solo alcune istruzioni sono eseguite senza manipolazione da parte del VMM, mentre le altre devono essere gestite in modo diverso: in pratica poiché alcune chiamate privilegiate non possono essere eseguite direttamente, devono essere preventivamente riscritte (si parla appunto di ***traduzione binaria***) dal VMM prima della loro esecuzione; si è riusciti a risolvere questo problema solo dagli anni 2005/06, attraverso le **tecniche di *trap-and-emulate*** ("intercetta ed emula") delle suddette istruzioni.

I software di full virtualization operano dunque in due modi*: alternando la tecnica di traduzione binaria all'esecuzione diretta*. Il flusso delle istruzioni viene analizzato dal programma di virtualizzazione alla ricerca di particolari istruzioni critiche (privilegiate); queste vengono modificate così da poter essere eseguite dal VMM con i corretti privilegi: la macchina virtuale non può operare in kernel mode ma deve lasciare che il VMM agisca per essa (cfr. *Fig.4.5*).

In sostanza ***la macchina virtuale viene eseguita su un interprete invece che direttamente sulla CPU; in questo modo l'interprete può risolvere i problemi dovuti ad operazioni che ostacolano la virtualizzazione, senza bisogno*** come detto ***di alcuna modifica al codice del kernel o delle applicazioni guest.***

Purtroppo la traduzione binaria ***peggiora le performance***, che soprattutto durante un'attività intensa di I/O, sono generalmente tra 80-97% di quelle dell'Host.
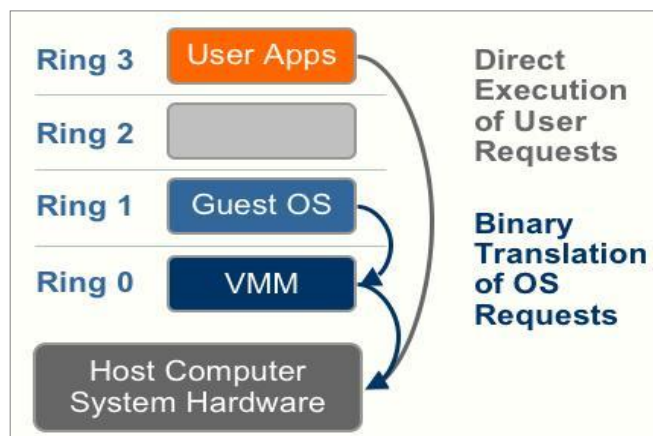
Fig. 4.5 – FULL VIRTUALIZATION

Esempi di prodotti che utilizzano questo tipo di virtualizzazione sono VMware Workstation, VMware Server e VirtualBox.

B – Paravirtualizzazione

"Para-" è un prefisso di origine greca che vuol dire *"accanto"*, *"con"* o *"a fianco"*. La paravirtualizzazione si riferisce infatti alla comunicazione tra il sistema operativo guest e l'Hypervisor per migliorare le prestazioni e l'efficienza del sistema.
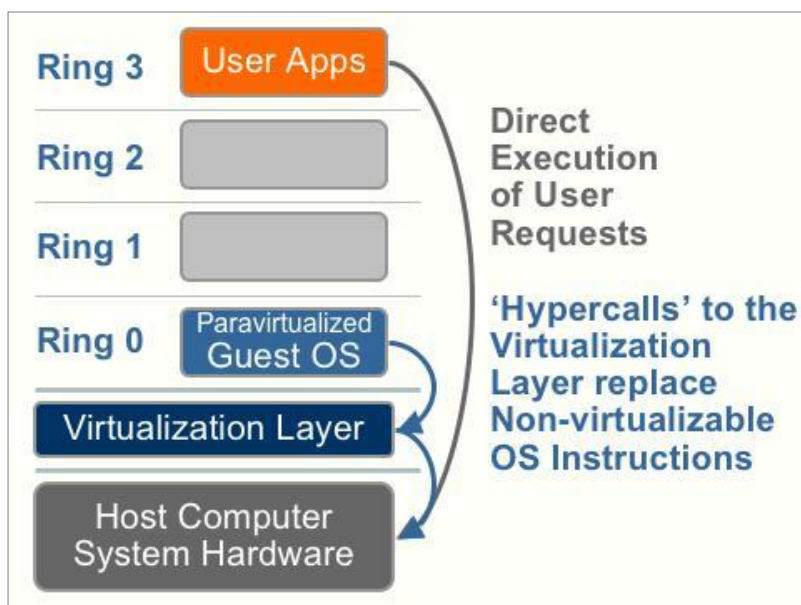
Fig. 4.6 – PARAVIRTUALIZZAZIONE

Questa tecnica di virtualizzazione prevede la ***simulazione parziale dell'hardware*** sottostante. Come mostrato in *Fig.4.6*, **comporta la modifica del kernel del sistema operativo guest** per sostituire le istruzioni non virtualizzabili con delle hypercalls che comunicano direttamente con il livello di virtualizzazione hypervisor. L'hypervisor fornisce anche interfacce hypercall per altre operazioni critiche del kernel quali la gestione della memoria e degli interrupt.

La paravirtualizzazione è diversa dalla full virtualization, in cui il sistema operativo non modificato non sa di essere virtualizzato e le chiamate del S.O. sensibili sono intrappolate ("trapped") con la traduzione binaria.

VANTAGGI: il pregio della paravirtualizzazione sta nel ***basso overhead*** di

virtualizzazione, ma il guadagno in termini di prestazioni della paravirtualizzazione rispetto alla virtualizzazione di tipo full può variare notevolmente a seconda del carico di lavoro; inoltre, in generale, mentre costruire il sofisticato supporto della traduzione binaria necessario per la virtualizzazione completa è molto difficile, modificare il sistema operativo guest per consentire la paravirtualizzazione è -almeno per certi sistemi operativi- relativamente facile.

SVANTAGGI: il limite maggiore della paravirtualizzazione è che essa ***non può sostenere sistemi operativi proprietari non modificabili*** (ad esempio Windows 2000/XP/Seven), e pertanto la sua compatibilità e portabilità è molto limitata. La paravirtualizzazione può anche introdurre importanti problemi di supporto e di manutenibilità negli ambienti di produzione in quanto richiede profonde modifiche al kernel del sistema operativo.

Il progetto open source Xen è un esempio di paravirtualizzazione che virtualizza processore e memoria utilizzando una versione modificata del kernel Linux e virtualizza l'I/O usando driver di periferica del sistema operativo guest personalizzati.

C – Virtualizzazione del Sistema Operativo

Questo tipo di virtualizzazione viene realizzata attraverso la creazione di copie del S.O. installato sull’Host. ***I sistemi guest risultano così a tutti gli effetti istanze del S.O. host***, con un proprio file system, librerie, applicazioni, ed un proprio utente di root. Essendo i sistemi operativi delle macchine guest equivalenti a quello della macchina host, ***le istanze guest non richiederanno un kernel privato, ma utilizzeranno lo stesso con un conseguente minor utilizzo di memoria fisica***.

VANTAGGI: tende ad essere ***leggera ed efficace***, viene eseguita a velocità normale (senza rallentamenti) e supporta tutto l'hardware nativo e le caratteristiche del sistema operativo per cui la macchina host è configurata.

SVANTAGGI: ***non supporta*** l'hosting di ***sistemi operativi misti***, come Windows e Linux contemporaneamente e, proprio per la condivisione del kernel, le ***istanze guest non sono così isolate*** e sicure come lo sono negli altri tipi di

virtualizzazione cosicché un problema di stabilità o di performance di un solo guest può influire negativamente sugli altri.

D – Virtualizzazione Hardware Assisted (o Nativa)

La *virtualizzazione hardware assistita* o ibrida, è la più recente tecnologia di virtualizzazione per architetture x86.
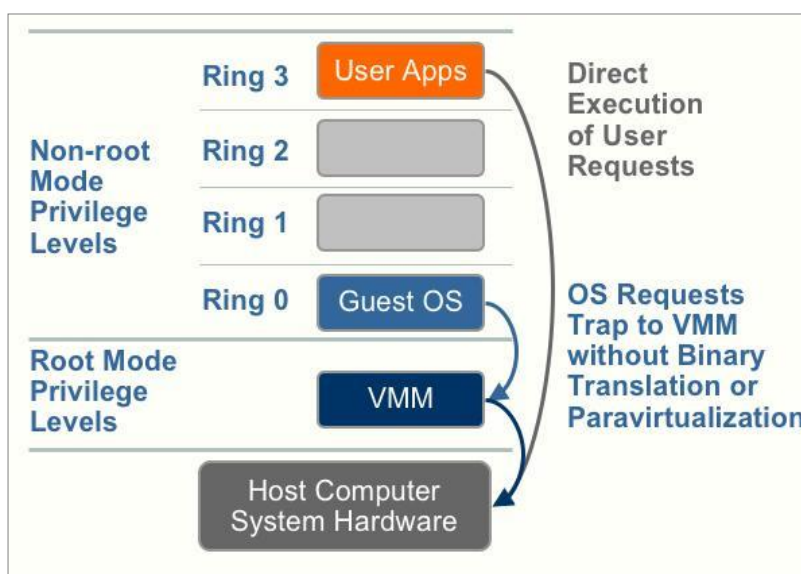
Fig. 4.7 – HARDWARE ASSISTED VIRTUALIZATION

I miglioramenti di prima generazione includono la Intel Virtualization Technology (VT-x) e la AMD-V di AMD le quali segnano le istruzioni privilegiate con una ***nuova modalità di esecuzione della CPU che consente di eseguire il VMM in una nuova modalità root sotto il ring 0***. Come illustrato in *Fig.4.7*, le chiamate privilegiate e sensibili sono impostate in modo che vengano *automaticamente intrappolate dall'hypervisor*, *eliminando la necessità della traduzione binaria o della paravirtualizzazione*. Lo stato guest viene memorizzato nelle Virtual Machine Control Structures (VT-x) o nelle Virtual Machine Control Blocks (AMD-V).

I processori con Intel VT-x e AMD-V sono disponibili dal 2006, per cui solo i sistemi più recenti contengono queste caratteristiche hardware assistite.

VANTAGGI: questo tipo di virtualizzazione ***è una sorta di "evoluzione***

hardware" della full, in quanto implementa in hardware alcune delle operazioni di traduzione binaria delle istruzioni effettuate dai VMM di tipo full; impiega in modo selettivo le tecniche di accelerazione per la memoria e le operazioni di I/O. Supporta inoltre i sistemi operativi x64 ed *ha le* ***più alte prestazioni*** *di processore, memoria e I/O fra tutti i tipi di macchine virtuali per architetture x86*.

SVANTAGGI: per funzionare ***richiede un processore che abbia l'accelerazione hardware assistita*** (Intel VT-x o AMD-V), che consente al sistema operativo ospite di avere accesso diretto alle risorse della piattaforma senza dover condividere il controllo dell'hardware o senza che venga emulato come faceva il VMM in precedenza. Può richiedere alcune modifiche (eventualmente limitate all'installazione di componenti aggiuntivi) ai S.O. guest, ma in maniera molto minore che nella paravirtualizzazione.

Quasi tutti i VMM inizialmente concepiti per la Full Virtualization traggono vantaggio dalla presenza delle nuove istruzioni (INTEL VT-x e AMD-V) implementando, in presenza di processori dotati delle suddette istruzioni, tecniche della Hardware Virtualization (un esempio in tal senso è appunto il VMM VirtualBox che, pur funzionando anche con processori che non hanno le suddette estensioni, trae notevole vantaggio specialmente in termini di efficienza e prestazioni dalla presenza delle stesse).

5.1 - UN ESEMPIO DI VMM TYPE-2: PRESENTAZIONE DI VIRTUALBOX

In conclusione di questo breve seminario sulla virtualizzazione, vogliamo fare un cenno ad **un esempio pratico di VMM** introducendo rapidamente **VirtualBox®**. Questo software di virtualizzazione vede la luce nel febbraio 2007 come prodotto di InnoTek GmbH, una software house tedesca acquisita l'anno successivo da Sun Microsystem a sua volta inglobata, nel 2010, da Oracle Corporation, attuale sviluppatrice di VirtualBox; tale VMM è rilasciato nella versione OSE (Open Source Edition) in licenza GNU General Public License (GPL) ed è pertanto liberamente utilizzabile in modo gratuito; questo fattore, oltre ad un livello di evoluzione tecnica che non fa troppo rimpiangere altri prodotti commerciali di fascia alta e generalmente assai più costosi (come ad es. VMware) rende questo tipo di Virtual Machine Manager particolarmente adatto ad una interpretazione della Virtualizzazione in chiave PMI (o comunque di qualsiasi realtà medio piccola e soprattutto con budget limitato).
Rifacendoci alle classificazioni precedentemente discusse, ***VirtualBox si configura come un virtualizzatore di Tipo2*** (che si esegue quindi nel contesto di un Sistema Operativo host) in grado di mettere in campo una ***virtualizzazione "full"*** sia di tipo esclusivamente *software* che -laddove il/i processori a disposizione la supportino- di tipo ***hardware-assisted***. A queste caratteristiche VirtualBox ne aggiunge un'altra decisamente interessante per un VMM di Tipo2: quella di essere ***cross-platform*** ovvero di poter girare su molteplici e differenti sistemi operativi host.
Tutto ciò, in pratica, significa che VirtualBox è un software di virtualizzazione che può essere installato su un computer della famiglia x86 (sia questo basato su processore Intel che su processore AMD) indipendentemente dal fatto che questo sia equipaggiato con un S.O. Windows (nella maggior parte delle sue versioni), piuttosto che Linux (nelle più diffuse distribuzioni), oppure Mac o ancora Solaris.
Inoltre le varie VM create con VirtualBox possono a loro volta eseguire una grande varietà di sistemi operativi (a 16 o 32 bit).

Pertanto VirtualBox si presenta come un VMM decisamente versatile in grado non solo di far funzionare VM con i più disparati S.O. ma anche, caratteristica assai interessante, di far girare la medesima macchina virtuale su piattaforme fisiche molto differenti fra loro (caratteristica tipicamente indicata con il termine di *portability*).

Inoltre, poiché come già detto VirtualBox è un virtualizzatore di tipo "full", tutti i suddetti Sistemi Operativi guest possono essere installati senza bisogno di alcuna modifica (cosa del resto necessaria per poter far girare senza problemi tutti i s.o. Microsoft, tipicamente proprietari e non aperti per nessun tipo di modifiche di terze parti).

Dopo aver discusso di una delle principali caratteristiche di VirtualBox, ovvero appunto la *portabilità*, vediamo di passare in rassegna alcune altre importanti caratteristiche del virtualizzatore scelto:

- ✓ **Virtualizzazione Hardware non necessaria:** sebbene VirtualBox sia in grado di sfruttare con profitto la presenza di processori che supportano le tecnologie Intel VT-x ovvero AMD-V, in molti scenari tali set di istruzioni non sono indispensabili al suo funzionamento; questo fa sì che VirtualBox possa essere impiegato con successo anche in presenza di hardware un po' più obsoleto che non presenta questo tipo di estensioni al set di istruzioni del processore.
- ✓ **Possibilità di aggiungere "Guest Addition" ai S.O. guest:** sebbene come detto i sistemi operativi installati nelle VM realizzate con VirtualBox non richiedano alcuna modifica, sono stati messi a punto -per la maggior parte dei S.O. guest supportati- delle estensioni dette *guest addition* che, installate nel S.O. della VM come un qualsiasi software di terze parti, ne migliorano le performance offrendo comunicazione ed integrazione aggiuntiva con il sistema host; ad esempio dopo l'installazione dell'apposita *guest addition* una VM supporterà l'aggiustamento automatico della risoluzione video, piuttosto che l'accelerazione grafica 3D ed altre numerose funzioni, come ad es. il supporto alle *"shared*

folders” ovvero *cartelle condivise fra guest e host*, sulle quali torneremo in un successivo paragrafo.

- ✓ **Grande supporto hardware per i S.O. guest:** tra gli altri VirtualBox supporta:
  - **Guest multiprocessing:** VirtualBox è in grado di presentare fino a 32 CPU virtuali a ciascuna VM, a prescindere dal numero di CPU e di core fisicamente presenti sull'host utilizzato.
  - **Supporto a dispositivi USB:** VirtualBox implementa un controller USB virtuale e permette di connettere qualsiasi periferica/dispositivo USB alle VM senza la necessità di installare alcun driver sull'host
  - **Compatibilità hardware:** VirtualBox virtualizza una vasta schiera di dispositivi, tra cui molti di quelli tipicamente utilizzati da altre piattaforme di virtualizzazione (ad es.: VMware); tra questi troviamo controller IDE, SATA e SCSI, diversi tipi di schede di rete virtuali, o schede audio, e ancora porte seriali e parallele virtuali, e molte altre periferiche che si trovano nell'hardware dei più moderni PC. Tutto ciò semplifica notevolmente la realizzazione di cloni virtuali a partire da immagini di macchine reali, nonché l'importazione in VirtualBox di macchine virtuali realizzate con prodotti di virtualizzazione di terze parti.
  - **Pieno supporto ACPI:** lo standard ACPI (Advanced Configuration and Power Interface) è pienamente supportato da VirtualBox; questo, oltre a semplificare l'importazione di cloni di macchine reali o di VM realizzate con altri virtualizzatori, permette inoltre a VirtualBox di riportare ai S.O. guest compatibili con gli standard ACPI lo stato energetico dell'host, segnalando ad esempio, nel caso di sistemi a batteria, la carica residua disponibile.
  - **Supporto per schermi multipli:** le VM realizzate con VirtualBox supportano risoluzioni anche superiori a quella dello schermo fisico dell'host, permettendo di suddividere l'output delle VM su

più schermi fisici distinti collegati all'host.

- ✓ **Possibilità di creare Istantanee delle VM:** VirtualBox può creare e salvare delle istantanee (o snapshots) che fotografano lo stato di una VM, e successivamente riportare l'ambiente della VM a tale stato, ripartendo magari da quella situazione per creare una configurazione alternativa della VM stessa; in questo modo il VMM può creare -per ciascuna VM- un vero e proprio "albero delle istantanee" permettendo di riportare, semplicemente ed in qualsiasi momento, lo stato di una VM ad una qualsiasi di queste situazioni precedenti. Da notare che si possono creare e cancellare gli snapshots di una VM mentre questa è in esecuzione (quindi in modo del tutto trasparente per il S.O. guest della stessa).
- ✓ **Accesso Remoto alle VM integrato:** VirtualBox implementa il VRDE (*VirtualBox Remote Desktop Extension*) che consente un accesso remoto con elevate performance ad ogni macchina virtuale; il VRDE è pienamente compatibile con il protocollo RDP di Microsoft, ma anziché sfruttare quest'ultimo è implementato direttamente nello strato di virtualizzazione del VMM; di conseguenza lavora anche con S.O. guest diversi da Microsoft Windows, e non richiede il supporto di specifiche applicazioni all'interno delle Macchine Virtuali; il VRDE fornisce inoltre speciali caratteristiche aggiuntive al tradizionale RDP, come ad esempio quella dell'USB over RDP, che permette di collegare in locale una qualsiasi periferica USB ad una VM che è in esecuzione remota su un server RDP di VirtualBox.

5.2 - PRIMI PASSI CON VIRTUALBOX®

Senza pretendere di sostituirci al manuale d'uso [12] del virtualizzatore VirtualBox, riteniamo comunque interessante, in questa parte conclusiva della trattazione fatta, fornire alcuni elementi di base per l'utilizzo del VMM scelto come esempio operativo e presentato nel paragrafo precedente.

Pertanto, nei prossimi paragrafi cercheremo di illustrare in modo sia pure sintetico i primi passi attraverso i quali realizzare con successo delle macchine virtuali con VirtualBox.

5.2.1 – Installazione e avvio di VirtualBox

VirtualBox è distribuito in differenti pacchetti, a seconda del tipo di sistema host su cui deve essere installato; l'installazione non presenta particolari problemi, ricalcando, per ciascun S.O. host, quella di un tipico software di terze parti; nel caso particolare di Sistemi host Windows, ad esempio, si presenta come il classico eseguibile di setup.

Una volta terminata l'installazione e lanciando per la prima volta VirtualBox ci si troverà di fronte ad una finestra come quella in *Fig.5.1*, che costituisce il cosiddetto *Gestore di VirtualBox* (o *VirtualBox Manager*) ovvero l'interfaccia grafica del VMM:
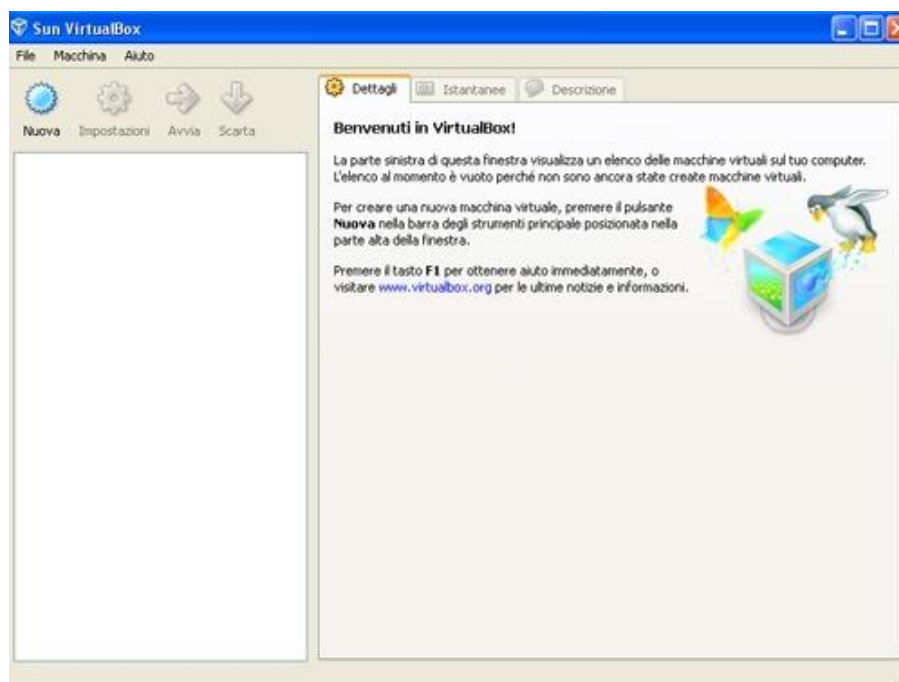
Fig. 5.1 – Primo avvio dell'interfaccia grafica del VMM VirtualBox

Sulla sinistra si può notare la sezione nella quale saranno successivamente elencate tutte le macchine virtuali disponibili sul sistema host (poiché al primo avvio non si è ancora creato alcuna VM, in figura la lista appare vuota).

Nella parte superiore della finestra una riga di pulsanti permetta di creare nuove VM e di interagire con quelle esistenti; nella parte destra della finestra verranno mostrate le proprietà della VM via via selezionata nella lista di sinistra (ancora una volta, poiché la figura sopra si riferisce ad una situazione in cui non sono ancora state create macchine virtuali, la sezione di destra contiene soltanto un generico messaggio di benvenuto).

Tanto per avere un'idea di come può apparire l'interfaccia grafica del VMM di VirtualBox dopo che si sono create alcune VM, si può considerare la *Fig.5.2*.
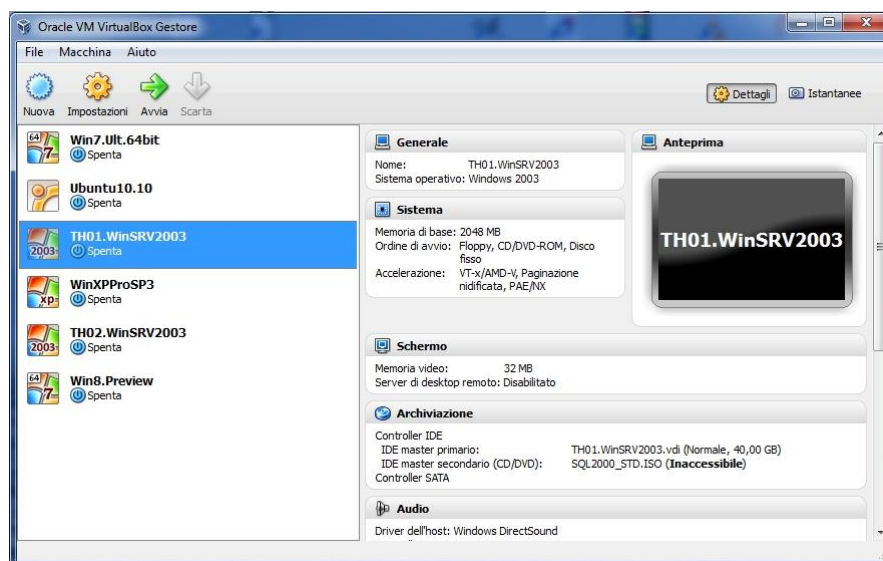
Fig. 5.2 – Tipico Aspetto della finestra del Gestore di VirtualBox in presenza di diverse VM

5.2.2 – Creazione di una VM con VirtualBox

Cliccando sul pulsante 'Nuova' (in alto a sinistra nella finestra del Gestore di VirtualBox, nelle figure precedenti) si apre una procedura guidata (cfr. *Fig.5.3*) che ci accompagna attraverso i passaggi fondamentali per la creazione di un VM chiedendoci una serie di informazioni minime necessarie alla sua creazione, ovvero:

- **Nome** della VM e **tipo di S.O.**: il nome è quello che vedremo comparire nella parte sinistra della finestra del VirtualBox Manager all'interno della lista delle VM presenti, ed ha pertanto solo scopo informativo, mentre il tipo di S.O. (da scegliere in un apposito elenco presentato dalla procedura

guidata) fa sì che, già in fase di creazione, il VMM abiliti o disabiliti certe caratteristiche della VM per renderla il più adatta possibile alla successiva installazione del tipo di S.O. guest qui indicato; per questo motivo -anche se tutti i parametri della VM creata possono essere rivisti e variati in un secondo momento- è utile in questa fase specificare il giusto tipo di S.O. che si intende installare.

- L'**ammontare di memoria RAM** (cfr. *Fig.5.4*) della VM: questo è un parametro fondamentale della macchina virtuale, in quanto ogni volta che viene avviata il VMM VirtualBox allocherà questo quantitativo di RAM
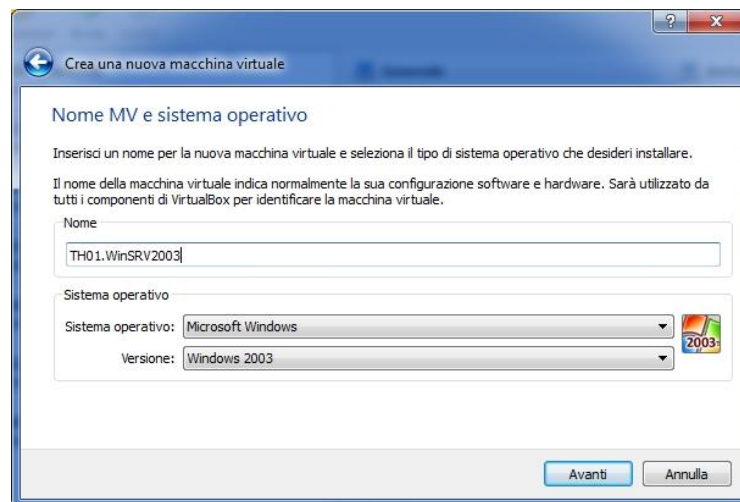
Fig. 5.3 - Creazione Nuova VM (definizione Nome e S.O.)

del sistema host presentandola al sistema operativo guest della VM come la RAM della macchina virtuale. È facile capire che questo parametro dovrà rispondere, caso per caso, a due contrapposte esigenze: da un lato dovrà essere sufficientemente elevato da permettere il corretto funzionamento della VM e di tutti i programmi che vorremo far girare su di essa, dall'altro dovrà essere sufficientemente contenuto da permettere di lasciare un quantitativo di RAM sufficiente alle necessità del Sistema host, oltre che, eventualmente, di tutte le altre VM che vorremo far girare in contemporanea.
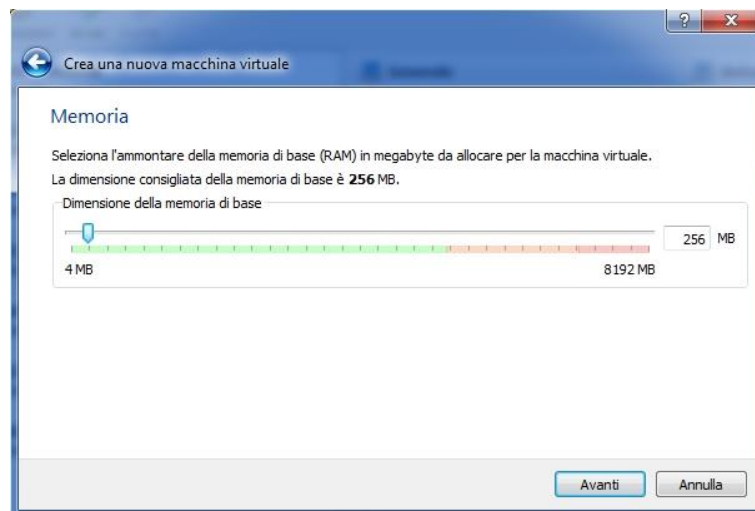
Fig. 5.4 - Creazione Nuova VM (impostazione RAM)

- Il passi successivi permettono di specificare un **hard disk virtuale** per la VM: ci sono diversi modi anche molto complicati con i quali VirtualBox può fornire una VM di spazio disco (e sui quali torneremo in seguito) ma quello più semplice consiste nell'utilizzare un grosso file immagine sull'hard disk fisico del sistema host, presentandone il contenuto alla VM come se fosse un disco completo. Da notare che questo file rappresenta un hard disk completo, e potrà quindi essere anche eventualmente copiato e spostato su un altro host con VirtualBox per essere impiegato con un'altra VM, oppure anche con una differente VM nell'ambito dello stesso host.
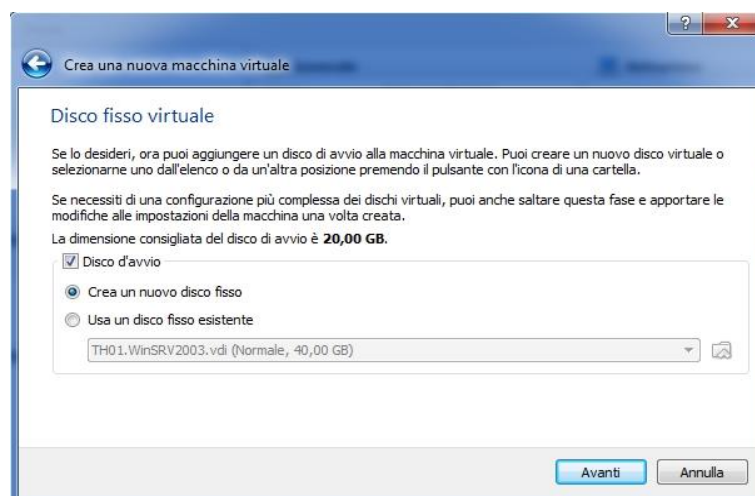
Fig. 5.5 - Creazione Nuova VM (Creazione Hard Disk)

Nella finestra in *Fig.5.5* è possibile, appunto, scegliere un hard disk virtuale precedentemente creato oppure crearne uno nuovo; in questo secondo caso ci viene chiesto, con un'ulteriore finestra della procedura guidata, che tipo di gestione si vuole adottare per il file di immagine dell'hard disk scegliendo fra le due supportate da VirtualBox:

a. un **file allocato dinamicamente**: inizialmente piccolo e che crescerà di dimensioni solo man mano che il guest immagazzinerà dati sul suo disco virtuale

b. un **file a dimensione fissa**: avrà subito la dimensione massima prevista per il file a prescindere dal livello di riempimento del disco virtuale del guest; questa seconda scelta determina a fronte di una maggiore occupazione di spazio sul disco fisico dell'host un certo incremento nelle prestazioni del disco virtuale del guest, perché riduce l'impatto sulle operazioni di gestione del VMM (ovvero il cosiddetto *overhead*).

Infine, per prevenire il completo esaurimento dello spazio fisico sul disco dell'host, VirtualBox ci chiede (cfr. *Fig.5.6*) di imporre un limite alla dimensione massima del file di immagine, che dovrà comunque essere scelta sufficientemente grande da contenere il Sistema Operativo guest e tutte le applicazioni che vorremo installare sulla VM (tipicamente per un moderno sistema Windows serviranno svariati gigabytes):
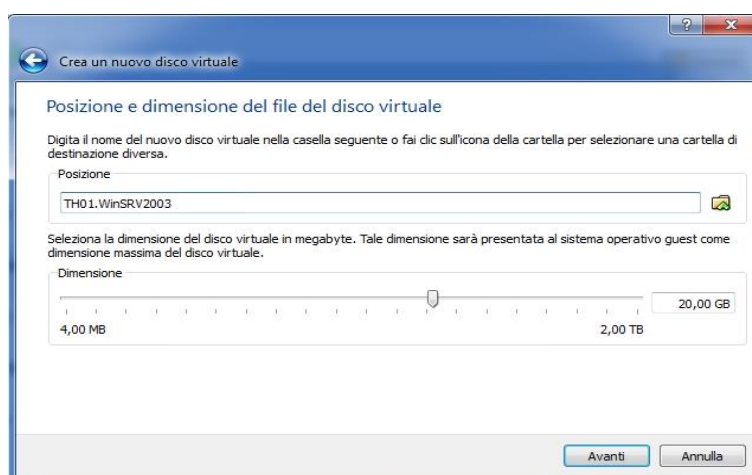
Fig. 5.6 - Creazione nuova VM (Dimensionamento Hard Disk)

Impostata la dimensione massima, e concludendo la procedura guidata si

ottiene la creazione della nostra nuova macchina virtuale, che comparirà quindi, con il nome da scelto, nella lista di sinistra del Gestore di VirtualBox.

5.2.3 – Esecuzione di una VM creata con VirtualBox

Una volta creata una macchina virtuale, essa comparirà nella lista delle VM disponibili nella finestra del VirtualBox Manager (cfr. *Fig.5.2*); per avviarla sarà sufficiente farci doppio clic oppure selezionarla e cliccare sul pulsante 'Start' della suddetta finestra.

Questo provocherà l'apertura di una nuova finestra nella quale si avvierà la VM scelta; tutto quello che vedremmo normalmente sullo schermo di una macchina reale, sarà mostrato nella finestra della nostra VM.

Poiché la VM è creata vuota, al suo primo avvio si comporterebbe esattamente come un computer reale senza sistema operativo: non farebbe assolutamente niente se non mostrare un messaggio di errore per la mancanza di sistema operativo. Per questo motivo al primo avvio di una VM si scatena un'apposita procedura guidata che ci aiuta a scegliere un supporto per l'installazione del S.O. guest: generalmente questo sarà un supporto fisico (CD o DVD), ed in tal caso VirtualBox configurerà la VM per utilizzare il lettore (o uno a scelta dei lettori, nel caso siano più di uno) dell'host, oppure potrà essere un'immagine ISO di un CD o DVD di installazione, che sempre VirtualBox provvederà a montare presentandola alla VM come un CD o DVD da cui fare il boot. In entrambe i casi dopo le opportune scelte nella procedura guidata si sarà in grado di installare il S.O. guest nella VM esattamente come si farebbe con quello di una macchina fisica.

Una volta installato e opportunamente configurato il S.O. guest sulla VM, sarà possibile utilizzare la macchina virtuale sostanzialmente come si farebbe con una analoga macchina reale, con la sostanziale differenza che essa verrà eseguita dal sistema host (in una finestra o volendo anche a schermo intero) in concomitanza con l'esecuzione di tutti gli altri processi e applicativi del S.O. host, nonché eventualmente di altre VM; l'unico limite pratico sarà costituito dalle risorse

hardware del sistema host (principalmente RAM e processore), e sarà appunto il VMM VirtualBox a gestire ed ottimizzare la condivisione delle stesse fra il sistema host ed i vari sistemi guest delle VM contemporaneamente in esecuzione.

5.2.4 – Spegnimento e Salvataggio dello Stato di una VM

Una qualsiasi VM in esecuzione su di un Sistema host, dovrebbe venir spenta secondo le modalità raccomandate dal particolare S.O. guest che vi è stato installato: nel caso di sistemi Windows, (ma il discorso è analogo anche per la maggior parte delle distribuzioni Linux ed in generale per molti S.O.), il corretto spegnimento del sistema passa attraverso l'apposita procedura di shutdown prevista dal S.O. stesso; seguendo tale procedura la VM inizia le operazioni di spegnimento, che possiamo direttamente seguire all'interno della finestra che ne contiene l'output, in modo del tutto simile a quanto avviene per una macchina reale fino al vero e proprio spegnimento, che nel caso della VM coincide con la chiusura della finestra nella quale era in esecuzione (oltre che nel rilascio delle risorse, tipo RAM e processore, che tornano così pienamente disponibili per il sistema host).

Tuttavia, rispetto al caso di macchine reali, le VM (che nel caso di VirtualBox sono sostanzialmente delle applicazioni in esecuzione sul server Host) possono venire arrestate anche in altri modi, ad esempio cliccando banalmente sul bottone “Chiudi” posizionato, come sempre, nell'angolo in alto a destra della loro finestra; in questo caso, VirtualBox ci presenterà tre possibili opzioni per la VM in questione, mostrate in *Fig.5.7*.

Fig. 5.7 – Opzioni allo spegnimento di una VM

La scelta fra le tre opzioni è cruciale; il loro significato è:

- **Salvare lo Stato della Macchina:** con questa opzione VirtualBox “congela” la VM salvando completamente il suo stato sul disco locale; ad un successivo riavvio della macchina virtuale, la ritroveremo esattamente nelle condizioni in cui si trovava con i vari programmi aperti e intenti a svolgere le medesime operazioni. In pratica questa opzione è l'equivalente per una VM della sospensione di un PC portatile (ad esempio in concomitanza della chiusura del coperchio).
- **Invia il Segnale di Arresto:** questa opzione determinerà l'invio di un segnale di shutdown ACPI alla VM, esattamente come avverrebbe in un PC reale premendo il pulsante di avvio/spegnimento. Nel caso in cui il S.O. guest installato nella VM sia compatibile con gli standard ACPI (cosa generalmente vera per S.O. moderni) questo scatenerà il corretto meccanismo di shutdown nella VM.
- **Spegni la Macchina:** con questa opzione VirtualBox arresterà la VM, ma senza salvarne lo stato, né eseguirne la procedura di shutdown. Questa opzione equivale a scollegare brutalmente dalla rete elettrica un PC reale: al riavvio della VM il S.O. operativo guest potrà richiedere un check per la consistenza dei dati del proprio hard disk (virtuale). Si capisce quindi che questa terza opzione non dovrebbe mai venir utilizzata, in quanto causa potenziale di perdita di dati e danneggiamento del S.O. guest. Come unica eccezione, se la VM è dotata di un’*Istantanea* (di cui tratteremo meglio nel prossimo paragrafo) è possibile usare questa opzione per ripristinare rapidamente tale istantanea: solo in questo caso eseguire lo spegnimento della VM non rischia di comprometterne il corretto funzionamento futuro.

5.2.5 – Istantanee (o Snapshots)

Una delle caratteristiche più interessanti del VMM VirtualBox è quella di poter salvare delle **istantanee** delle VM; con tale termine si indica appunto una sorta di fotografia istantanea dello stato di una VM, utilizzabile per salvare una particolare situazione/configurazione della macchina virtuale che possa poi essere

utilizzata in un secondo momento per riportare la VM esattamente in quello stato, a prescindere da quali e quanti possano essere stati i cambiamenti e le modifiche operate sulla VM nel frattempo.

Le varie istantanee di una VM possono essere viste e gestite dalla solita finestra del Gestore di VirtualBox (cfr. *Fig. 5.8*) selezionando una macchina virtuale dalla lista di sinistra e cliccando sul bottone “Istantanee” in alto nella parte destra:
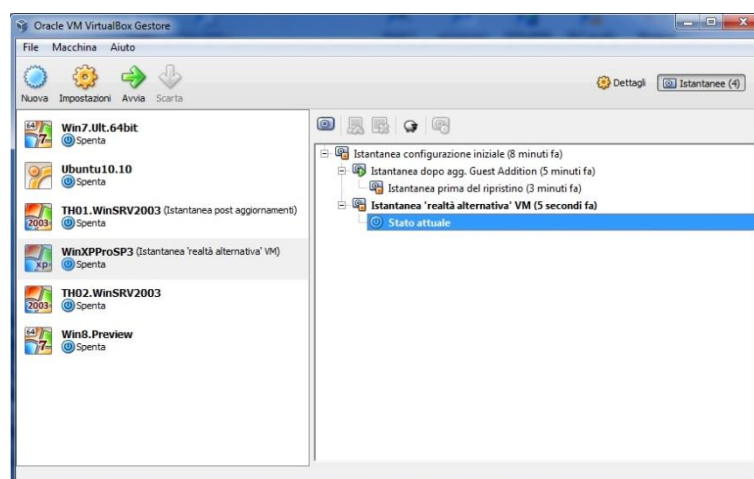
Fig. 5.8 - Gestione Istantanee

Inizialmente, fino a che non si realizzerà almeno un'istantanea di uno stato della VM, la lista sulla destra apparirà vuota, ad eccezione della voce relativa allo “Stato Attuale” che rappresenta, come si evince dal nome, il “presente” nella storia della VM.

Le **istantanee** di una VM possono essere **create** anche durante il funzionamento della stessa e vanno a popolare la lista delle istantanee della Fig. precedente, determinando uno spostamento della voce “Stato Attuale” al di sotto dell'ultima creata, ad indicare che lo stato corrente della VM è una variazione a partire dall'ultima istantanea catturata. Se successivamente si creano ulteriori istantanee si può vedere che esse saranno mostrate in sequenza ed ognuna sarà derivata dalla precedente.

VirtualBox permette di catturare un numero teoricamente arbitrario di istantanee, essendo l'unica limitazione pratica lo spazio disponibile sul disco dell'host: poiché ogni istantanea conserva lo stato del momento della VM è infatti chiaro che

occuperà un certo spazio sul disco.

Il **ripristino** di una qualsiasi **istantanea** della VM può essere fatto semplicemente cliccando con il bottone destro del mouse direttamente su di essa e scegliendo l'apposita voce nel menù contestuale; così facendo si torna indietro (o avanti) nel tempo: lo stato attuale della VM viene perso ed essa viene riportata esattamente allo stato in cui si trovava quando è stata creata l'istantanea selezionata.

E' importante notare che il ripristino di un'istantanea è un'operazione che ha effetto sull'intero stato della VM, ivi compresi gli hard disk (virtuali) ad esse connessi: questo significa che tutti i file creati o modificati dopo la creazione dell'istantanea che si vuole ripristinare verranno persi (per evitare la perdita di dati in tale situazione è possibile creare una nuova istantanea della VM prima di procedere al ripristino).

Ripristinando un'istantanea precedente allo stato attuale e successivamente creandone di nuova a partire da quella, è possibile creare una sorta di realtà alternativa per la VM, e passare successivamente dall'una all'altra storia della macchina virtuale. Il risultato può così essere quello di una sorta di albero delle istantanee (come appunto in *Fig.5.8*) con possibili percorsi alternativi nella *timeline* della VM.

Le **istantanee** di una VM possono infine anche essere **cancellate**: questo non avrà effetto sullo stato attuale della VM ma permetterà di rilasciare spazio su disco eliminando i file utilizzati da VirtualBox per memorizzare lo stato della VM relativo a tale istantanea. Si noti che poiché in realtà per ciascuna istantanea VirtualBox memorizza non l'intero stato del disco ma soltanto le differenze dall'istantanea immediatamente precedente nell'albero delle istantanee stesse, l'operazione di cancellazione potrebbe risultare di fatto più lunga e complessa di quelle di creazione e ripristino, potendo richiedere -a seconda dell'istantanea che si vuole eliminare- anche un consistente lavoro di copia di file ed informazioni di stato da un’istantanea all'altra.

A partire dalla versione 3.2 di VirtualBox le operazioni di cancellazione di una istantanea possono generalmente essere eseguite anche durante il funzionamento della macchina virtuale interessata; per quanto appena detto, tuttavia, possono

verificarsi casi per i quali la cancellazione di un'istantanea non è gestibile a VM funzionante: in tali situazioni un opportuno messaggio ci avviserà che la cancellazione dell'istantanea selezionata potrà avvenire solo a macchina virtuale spenta.

5.2.6 – Cancellazione di una VM

Per cancellare un VM precedentemente creata con VirtualBox, è sufficiente cliccarvi con il bottone destro del mouse nella lista delle VM che compare sulla destra della finestra del VirtualBox manager, scegliendo poi "Rimuovi" nel menù contestuale che appare. In questo modo compare una finestra di conferma che permette inoltre di sceglie se rimuovere solo la VM o cancellare anche definitivamente dal disco dell'host tutti i file ad essa associati.

Si noti che la cancellazione di una macchina virtuale è inibita durante il funzionamento della stessa.

5.2.6 – Realizzazione del Clone di una VM

Per poter effettuare dei test su possibili configurazioni diverse di una stessa VM, o anche semplicemente per effettuarne un backup, VirtualBox mette a disposizione un'utilissima procedura guidata che permette di clonare una macchina virtuale.

Tale procedura può essere richiamata sia dal menù contestuale delle VM nella solita lista del Gestore di VirtualBox, che dalla finestra delle istantanee relative alla VM in questione, come mostrato in *Fig.5.9*.
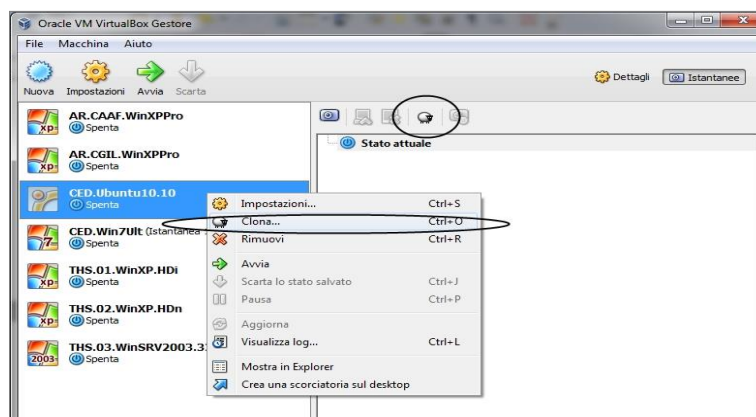
Fig. 5.9 – Clonazione di una VM

Come primo passo la procedura guidata ci chiede di scegliere un nome per la VM clone, ed inoltre ci permette di scegliere se reinizializzare il MAC address di tutte le schede di rete della VM (cfr. *Fig.5.10*).
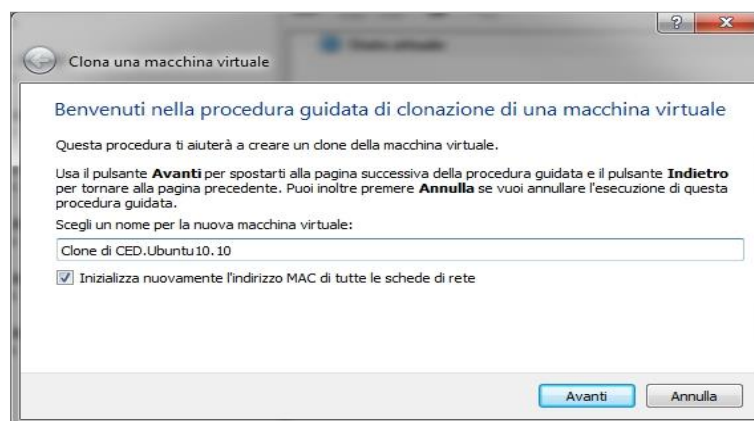
Fig. 5.10 - Clonazione di una VM (Procedura Guidata)

Questa opzione è particolarmente utile quando la VM sorgente e la VM clonata dovranno operare nel contesto della stessa rete.

Proseguendo con la creazione guidata ci viene richiesto se vogliamo realizzare un *Clone Completo* (Full Clone) o un *Clone Collegato* (Linked Clone): nel primo caso sarà creata una copia completa (incluso le immagini di tutti i dischi virtuali) della VM originale; nel secondo caso, invece, sarà creata una nuova VM ma le immagini dei dischi virtuali punteranno a quelle della VM originale (per la quale sarà creata una nuova istantanea di riferimento).

Si noti che per clonare una macchina virtuale è necessario che questa risulti spenta.

5.2.7 – Utilizzo di front-end alternativi al VirtualBox Manager

A conclusione del rapido excursus fatto per introdurre il VMM VirtualBox, è interessante notare che le VM create con tale virtualizzatore sono indipendenti dall'interfaccia grafica del VirtualBox manager fin qui trattata, che può essere anche chiusa una volta avviata la o le VM che ci interessano.

Non solo, tale interfaccia è solo la soluzione più semplice ed intuitiva per interagire con le macchine virtuali, ma non è l'unica possibilità, (né certamente

sempre la più indicata).

Tanto per fare un esempio è possibile avviare una VM mediante l'interfaccia grafica del Manager e poi chiudere il manager stesso ed arrestare la VM mediante riga di comando; oppure, utilizzando il supporto di VirtualBox per il Desktop Remoto (VRDP) di cui si è parlato, è possibile eseguire le macchine virtuali senza alcun output grafico sul server host (si parla in questo caso anche di headless server) e reindirizzare tutti gli output in remoto tramite la rete.

Più in dettaglio, i front-end disponibili per VirtualBox (tutti distribuiti nel pacchetto standard) sono:

1. Il **Gestore di VirtualBox** (o VirtualBox Manager), cioè l'interfaccia utente di tipo grafico cui si è sin qui fatto riferimento; tale front-end è senz'altro il più semplice da utilizzare almeno inizialmente, e permette la gestione e configurazione della maggior parte delle caratteristiche di VirtualBox.
2. **VBoxManage**: è l'interfaccia a riga di comando per l'automazione ed il controllo di ogni dettaglio ed aspetto di VirtualBox; è ovviamente più complessa da utilizzare (in quanto richiede l'apprendimento di una precisa sintassi dei comandi) ma permette di gestire aspetti delle VM altrimenti non controllabili dall'interfaccia grafica. Un elenco dei di VBoxManage è contenuto nell'Appendice A.
3. **VBoxSDL:** è un semplice front-end di tipo grafico (alternativo al più completo Gestore) volutamente semplificato e limitato dal punto di vista delle caratteristiche e delle possibilità di controllo delle VM; il suo scopo è unicamente quello di fornire un sistema decisamente meno pesante del Gestore completo per mostrare le VM controllate nel dettaglio mediante VBoxManage.
4. **VBoxHeadless:** è un altro front-end che non produce alcun output visibile sull'Host, ma si comporta semplicemente come server VRDP, rendendo disponibile l'accesso remoto via rete alle varie VM.

Tutti e quattro i tipi di front-end sopra citati possono essere usati indifferentemente in alternativa o anche in combinazione per regolare e gestire le varie VM.

Tipicamente le interfacce grafiche sono più indicate in una fase di studio preliminare e di creazione dei prototipi delle macchine virtuali e in generale dell'ambiente virtuale che si vuole costruire, in virtù della loro immediatezza che semplifica l'iniziale gestione delle varie VM.
Per contro si capisce che front-end più spartani e con meno fronzoli grafici, saranno inevitabilmente più prestanti e tipicamente più indicati per un utilizzo a regime in ambiente di produzione; proprio la presenza di un'interfaccia da riga di comando come appunto VBoxManage, avvicina di molto le prestazioni e le caratteristiche offerte da un prodotto gratuito come VirtualBox a quelle di prodotti commerciali più blasonati e costosi come VMware.

5.3 - CONFIGURAZIONE DELLE VM DI VIRTUALBOX

Nel paragrafo precedente abbiamo iniziato ad introdurre gli elementi fondamentali relativi a VirtualBox ed alle macchine virtuali realizzabili con tale virtualizzatore.
Come abbiamo visto, già in fase di creazione di una VM è possibile impostarne alcune caratteristiche tipo il quantitativo di RAM ovvero l'ammontare di spazio disco, ecc.; queste e molte altre caratteristiche relative all'hardware (virtualizzato) della VM possono essere impostate anche successivamente sia tramite l'interfaccia grafica offerta dal Gestore di VirtualBox, sia -in modo molto più fine e con possibilità di regolazione assai più ampie- mediante l'interfaccia a riga di comando di VBoxManage.
Le possibili regolazioni ed i settaggi disponibili per le VM realizzabili con VirtualBox sono numerosissimi e variamente complicati, e la loro completa descrizione esula e travalica l'obiettivo di questo breve seminario.
Di seguito, però, vogliamo considerare un po' più nel dettaglio almeno alcune delle principali configurazioni e regolazioni applicabili alle nostre macchine virtuali, con particolare attenzione a quelle adottate nella realizzazione dell'ambiente virtuale oggetto della tesi stessa.
Cominciamo dalle principali regolazioni operabili tramite interfaccia grafica,

rimandando alla Appendice A la trattazione dell'interfaccia a riga di comando VBoxManage e di alcuni dei suoi comandi.

Selezionando una VM nella lista del Gestore di VirtualBox, e cliccando sul pulsante Impostazioni, si accede alla finestra delle impostazioni della VM in questione (cfr. *Fig.5.11*).

Tale finestra è organizzata in *sezioni* (nella colonna di sinistra), ciascuna delle quali vede a propria volta le varie regolazioni ed impostazioni suddivise in una o più *schede* (che nel seguito indicheremo a volte anche come *tab* e che sono poste nella parte di destra della finestra stessa).

5.3.1 – Sezione *Generale*

Nella sezione Generale sono raggruppate le informazioni fondamentali della macchina virtuale, ripartite in tre *tab* o *schede*:
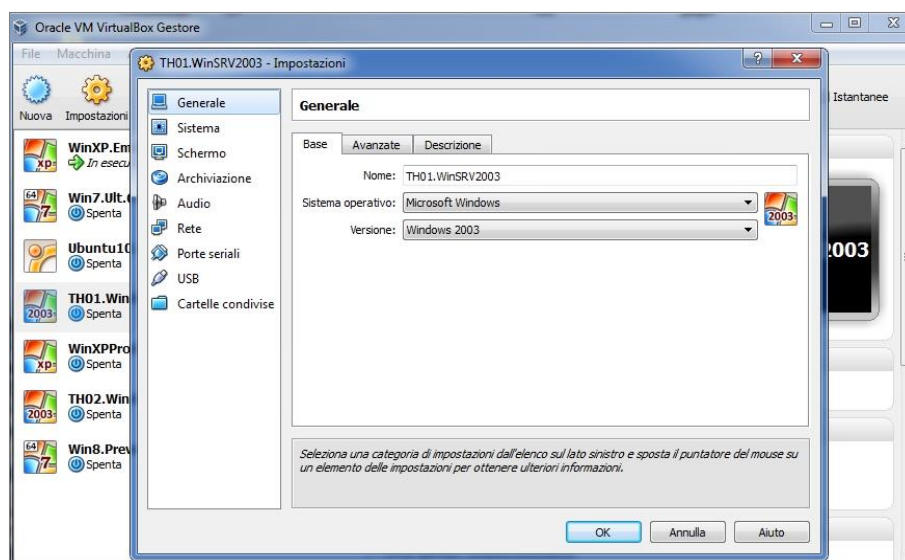
Fig. 5.11 – Finestra Impostazioni di una VM

- **Base:** In questa scheda è possibile impostare il *Nome* con il quale la VM è mostrata nella lista delle macchine virtuali, oltre che il tipo e la *Versione* di Sistema Operativo guest che è (o sarà) installato nella VM. Questo è il medesimo settaggio specificato nella procedura guidata di creazione di una nuova VM (cfr. *paragrafo 5.2.2*).
- **Avanzate:** in questa scheda (cfr. *Fig. 5.12*) si può scegliere il *percorso* -sul

disco del server host- *in cui verranno salvate eventuali istantanee* della macchina virtuale.
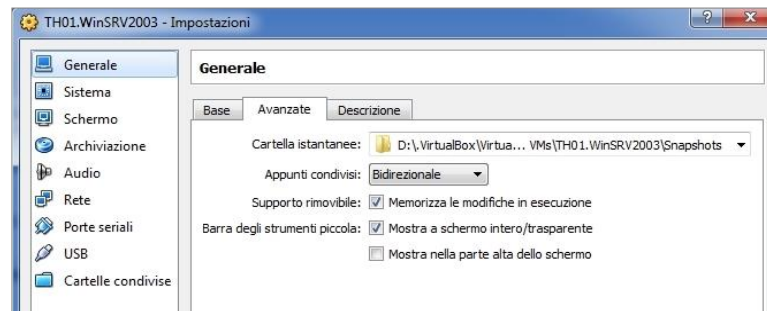
Fig. 5.12 - Impostazioni Avanzate

Sempre da questo tab è possibile decidere se la VM debba *condividere* o meno con l'host *gli appunti* e, nel caso, se la condivisione deve essere bidirezionale, oppure a senso unico (da guest a host o viceversa). Tale regolazione, per quanto comoda per passare dati con un semplice copia e incolla da host a VM ovvero da una VM all'altra, riduce, come si capisce, l'isolamento fra le singole VM e fra queste ed il server host.

- **Descrizione:** In questa scheda è possibile inserire una descrizione della macchina virtuale.

5.3.2 – Sezione *Sistema*

Nella sezione Sistema (cfr. *Fig.5.13*) trovano posto varie impostazioni relative all'hardware di base presentato alla VM; anche qui abbiamo tre schede:
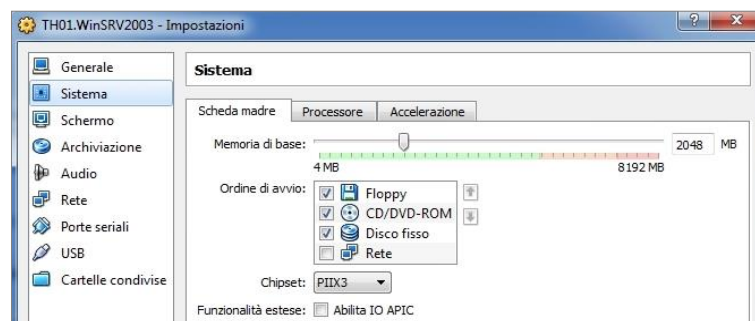
Fig. 5.13 – Impostazioni di Sistema (Scheda Madre)

- **Scheda Madre:** Qui è possibile settare l'ammontare di *memoria RAM* presentata alla VM, l'*Ordine di avvio* delle periferiche di boot (esattamente

come faremmo nel BIOS di una macchina reale), oltre al *Chipset* emulato (selezionabile fra il più consolidato PIIX3 ed il più moderno ICH9 supportato da alcuni S.O. più moderni come ad es. il Mac OS X server), o ad una serie di *Funzionalità Estese* tra le quali l'I/O APIC (Advanced Programmable Interrupt Controllers) che permettono di virtualizzare per la VM alcune recenti caratteristiche hardware della famiglia x86, richieste tipicamente dai S.O. a 64 bit.

- **Processore:** In questa scheda è possibile impostare quanti *Processori* (intesi come CPU-cores) debbano essere virtualmente presentate al Sistema Operativo guest della VM (cfr. *Fig.5.14*).
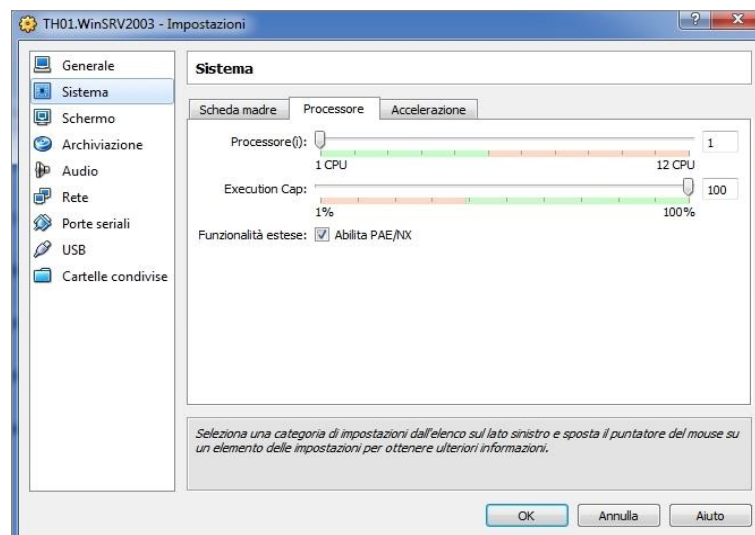
Fig. 5.14 – Impostazioni di Sistema (Processore)

È possibile teoricamente presentare fino a 32 CPU per ciascuna macchina virtuale, purché ovviamente non si superi il numero dei core fisicamente disponibili nell'host. Sempre in questa scheda è possibile anche fissare il cosiddetto *Execution Cap*, ovvero la percentuale di tempo che una CPU fisica dedica all'emulazione di una CPU virtuale: il default è il 100% (ovvero nessun limite) ma un'impostazione, ad esempio, del 50% significherebbe che ciascuna CPU virtuale ha a disposizione il 50% dei cicli macchina di una CPU reale. Si noti che sebbene limitare il tempo di esecuzione delle CPU virtuali appaia come un interessante ulteriore passo

nel segno del “time sharing” che è alla base, anche storicamente, della virtualizzazione, in pratica l'adozione di detta limitazione per la/le CPU di una VM può indurre problemi di timing nel sistema guest.

Nella medesima scheda, inoltre, l'impostazione *Enable PAE/NX* determina se le funzionalità PAE (Physical Address Extension) ed NX (No eXecute bit) eventualmente possedute dalla CPU dell'host debbano essere presentate o meno anche alla CPU della VM; queste caratteristiche se abilitate e supportate anche dal Sistema Operativo permettono tra l'altro di accedere a più di 4 GB di memoria anche a sistemi a 32 bit.

- **Accelerazione:** In quest'ultimo tab della sezione Sistema (cfr. *Fig.5.15*), è possibile decidere se abilitare o meno per la VM in questione la *Virtualizzazione Hardware* sfruttando le istruzioni estese Intel VT-x o AMD-V nonché l'utilizzo della *paginazione nidificata* della CPU dell'host. Nel caso in cui la CPU dell'host non supporti le estensioni per la
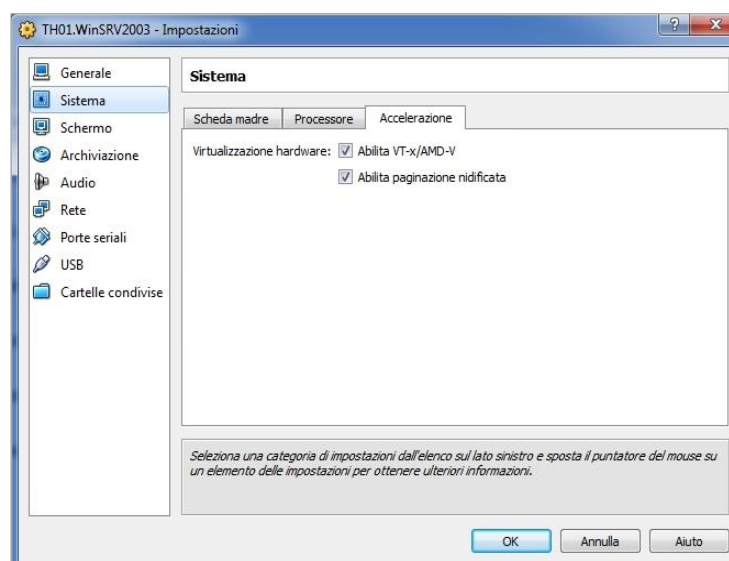
Fig. 5.15 – Impostazioni di Sistema (Accelerazione)

virtualizzazione hardware (cosa generalmente limitata alle CPU di fabbricazione antecedente il 2007) la scheda Accelerazione risulterà disabilitata.

Come già detto VirtualBox, a differenza di altri virtualizzatori, riesce a funzionare anche su sistemi host più datati con CPU prive delle istruzioni estese per la virtualizzazione hardware, sopperendo con la virtualizzazione

software; ovviamente laddove possibile l'abilitazione delle suddette istruzioni riduce il lavoro di overhead del virtualizzatore e determina un incremento anche significativo delle prestazioni delle VM.

5.3.3 – Sezione *Schermo*

Nella sezione Schermo trovano posto le impostazioni relative alle periferiche video (virtuali) della VM; anche tale sezione è organizzata in schede (cfr. *Fig.5.16*).
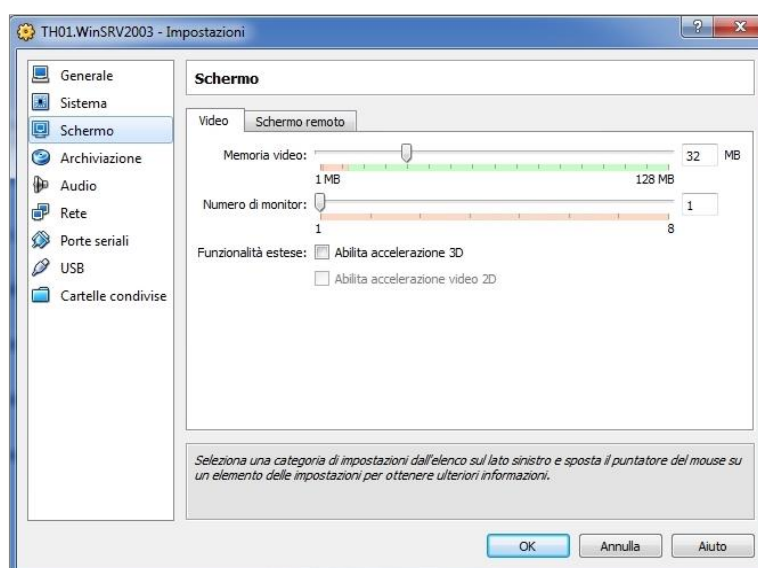
Fig. 5.16 – Impostazioni Schermo (Video)

- **Video:** Qui è possibile fissare la quantità di *memoria video* della scheda grafica (virtuale) presentata alla VM; così come per la memoria RAM della VM anche tale porzione di memoria sarà ritagliata da VirtualBox dalla memoria fisica dell'host. Così come per le schede video reali, ad un maggior quantitativo di memoria corrisponderà la disponibilità risoluzioni e profondità di colore più elevate per il S.O. guest. Sempre nello stesso tab è possibile scegliere il *numero di monitor* -fino ad un massimo di 8- virtualmente connessi alla VM (nel caso in cui il sistema operativo guest della VM supporti più monitor, come nel caso di Windows) così come pure abilitare l'accelerazione 2D e 3D per il guest (feature quest'ultime che richiedono comunque anche l'installazione delle già citate *Guest Addition*

nel S.O. guest della VM, cfr. *paragrafo 5.1*)

- **Schermo Remoto:** in questa scheda (cfr. *Fig.5.17*) è possibile abilitare per la VM il *VRDP* (VirtualBox Remote Display) *server* integrato in VirtualBox. Questo permette di connettersi da remoto alla macchina virtuale in questione utilizzando un qualsiasi rdp viewer standard, come *mstsc.exe* di Microsoft o l'open-source *rdesktop* per Linux o Solaris.
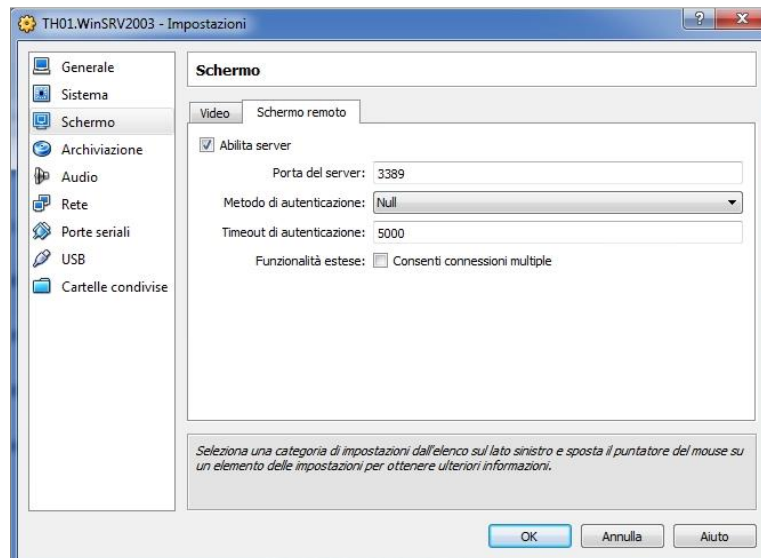
Fig. 5.17 – Impostazioni Schermo (Schermo Remoto)

L'*abilitazione del server* VRDP integrato in VirtualBox permette di svincolarsi dal server RDP del particolare S.O. guest installato sulla VM, e, specificando una *porta* diversa per diverse macchine virtuali, permette di accedere a più VM mediante lo stesso indirizzo IP (dell'host).

5.3.4 – Sezione *Archiviazione*

Nella sezione Archiviazione della finestra Impostazioni, è possibile definire nei minimi dettagli come e quali supporti di memorizzazione (reali e virtuali) sono connessi alla nostra macchina virtuale.

In un computer reale il collegamento fisico degli hard disk e dei lettori CD/DVD o Floppy è realizzato da componenti hardware che costituiscono i cosiddetti *controller dei dischi*; in modo del tutto analogo VirtualBox presenta alle VM dei

controller virtuali ai quali sono poi connessi i vari dispositivi di memorizzazione che possono a loro volta essere virtuali (immagini di hard disk, CD/DVD o floppy) o anche reali (come ad esempio i lettori CD/DVD o floppy dell'host, piuttosto che una SAN, ecc.).

Anche i *controller virtuali* sintetizzati da VirtualBox possono essere, così come quelli reali presenti fisicamente sulle motherboard dei computer, di vario tipo; alla versione attuale i controller previsti da VirtualBox possono essere di tipo IDE, SATA, SCSI e SAS. Ovviamente, come nel caso di sistemi fisici, non è sufficiente che la VM presenti un certo tipo di controller, ma è anche necessario che il S.O. guest installato su di essa lo supporti.

Come risultato della procedura guidata precedentemente descritta per la realizzazione di una macchina virtuale, la sezione Archiviazione della finestra Impostazioni della VM in questione avrà l'aspetto mostrato nella *Fig.5.18*, che rappresenta la tipica configurazione per i dispositivi di memorizzazione nel caso in cui il S.O. guest sia sufficientemente moderno da supportare nativamente sia
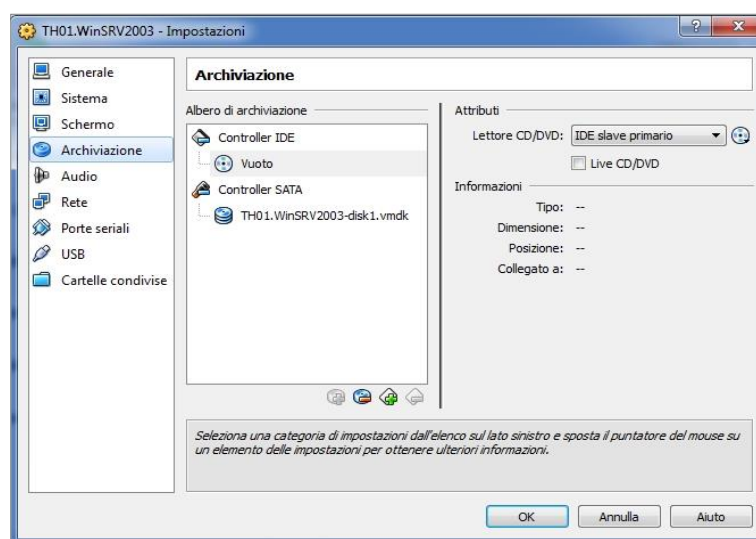
Fig. 5.18 – Impostazioni Archiviazione

controller IDE che SATA; in essa si possono vedere:

- un **controller IDE** al quale è connesso un CD/DVD virtuale (che può essere sia un'immagine che uno dei lettori ottici fisicamente presenti

sull'host)

- un **controller SATA** al quale è connesso un hard disk virtuale (o anche più di uno)

Nel caso in cui, all'atto della creazione della VM sia stato scelto un S.O. più vecchio (cfr. *paragrafo 5.2.2*), che non supporta i controller SATA senza il caricamento di driver aggiuntivi (come ad es. Windows XP) il layout standard della sezione Archiviazione sarà diverso, con il solo controller IDE al quale saranno attaccati sia il lettore CD/DVD che l'hard disk, e questo perché VirtualBox provvede alla realizzazione di un ambiente in cui tutti i device virtuali siano subito funzionanti e riconosciuti dal S.O. guest che dovremo installare sulla VM.

Naturalmente è possibile modificare le impostazioni della sezione Archiviazione di una VM, aggiungendo a piacimento controller (IDE, SATA,SCSI,SAS o floppy controller) ai quali poi è possibile connettere o disconnettere uno o più dischi o dispositivi; questo può essere fatto (fermo restando il disco sul quale si è installato il S.O. guest) anche dopo l'istallazione di sistema operativo e applicativi nella VM, ad esempio se si vuole montare come disco aggiuntivo il disco virtuale di un'altra VM da cui si debbono copiare dei dati.

Proprio i dischi virtuali che possiamo connettere ai rispettivi controller di una VM, meritano un piccolo approfondimento; come abbiamo visto (cfr. *paragrafo 5.2.2*) questi sono tipicamente file immagine fisicamente posti sul disco dell'host (o su un disco di rete) e possono avere dimensione fissa o dinamicamente variabile man mano che si riempiono.

Quello di cui vogliamo occuparci adesso riguarda la modalità di scrittura definibile per ciascuna immagine di hard disk virtuale, ovvero il modo in cui vi si riflettono le operazioni di scrittura da parte di una VM nonché le operazioni di creazione/ripristino delle istantanee (o snapshots).

Ci sono fino a sei modalità diverse definibili per l'immagine di un disco, e precisamente:

1. *Modalità Normale:* è l'impostazione di default, e con essa non ci sono

restrizioni sul modo in cui il guest può leggere da o scrivere sul disco. Quando si effettua una istantanea della VM (cfr. *paragrafo 5.2.5*) lo stato di un disco in 'modalità normale' viene salvato insieme all'istantanea stessa ed è pertanto completamente ripristinato (con la situazione di tutti i file e cartelle che contiene) nel momento in cui venga ripristinata l'istantanea della VM. Anche se è possibile montare la stessa immagine disco di tipo 'normale' su più macchine virtuali differenti, soltanto una di queste può essere eseguita simultaneamente, altrimenti si verificherebbero conflitti di scrittura sullo stesso disco.

2. *Modalità Writethrough*: al contrario del caso precedente, un disco in tale modalità non viene assolutamente riguardato dalla creazione e dal ripristino delle istantanee: questo può essere utile per il salvataggio di dati e documenti critici di cui non si vuole perdere la versione aggiornata in caso di ripristino di una vecchia istantanea della VM.
3. *Modalità Condivisibile*: come nel caso della Writethrough anche con questa modalità lo stato del disco non viene né salvato né ripristinato con eventuali istantanee della VM; la differenza fondamentale sta nel fatto che un disco in modalità 'condivisibile' può essere montato su diverse macchine virtuali che sono in esecuzione contemporaneamente: ovviamente per un corretto funzionamento è necessario che le macchine virtuali in questione siano equipaggiate con un apposito filesystem (di tipo cluster) e con applicazioni appositamente disegnate per l'utilizzo concomitante del disco.
4. *Modalità Invariabile*: un disco in tale modalità memorizza gli accessi in scrittura solo temporaneamente, ovvero per il tempo in cui la VM è accesa; tutti i cambiamenti vengono persi ad ogni nuovo avvio della macchina virtuale. Con questa modalità la stessa immagine disco può essere utilizzata contemporaneamente da diverse macchine virtuali (in pratica l'immagine invariabile serve solo da comune punto di partenza al boot delle VM, ma poi le eventuali modifiche al S.O. e ai file vengono salvate in immagini temporanee distinte durante il funzionamento delle varie VM;

tutte le VM, al successivo avvio ripartono sempre dalla stessa immagine immutabile perdendo memoria delle operazioni di scrittura della precedente sessione di funzionamento).

Creare un disco virtuale direttamente in modalità invariabile ha poco senso (perché lo ritroveremmo sempre completamente vuoto ad ogni avvio della VM) mentre invece è assai più interessante prendere un disco in modalità normale e -una volta che si ritiene riempito con tutto quello che serve- trasformarlo in invariabile, congelando di fatto la sua situazione (ad esempio per ritrovarsi sempre con una VM con S.O. guest “pulito” e fresco di installazione).

Nel caso in cui venga fatta un'istantanea di una VM con un disco in modalità invariabile, ad ogni successivo avvio della macchina il disco si resetterà a tale istantanea (anziché allo stato originario).

5. *Modalità a Collegamento Multiplo*: un disco in tale modalità può essere montato su più VM anche se queste sono in esecuzione contemporaneamente ed anche se hanno file system di tipo standard (che non prevedono accessi concorrenti allo stesso supporto). Infatti con questa modalità per ciascuna VM cui un tale disco è attaccato viene creata un'immagine differenziale: i dati scritti da ciascuna macchina virtuale non sono visibili per le altre, ed in pratica ogni VM crea la propria storia a partire dall'immagine a collegamento multiplo.

   Tecnicamente questa modalità differisce da quella invariabile solo per il fatto che le immagini differenziali non vengono eliminate ad ogni nuovo avvio.
6. *Modalità di Sola Lettura*: questa modalità è automaticamente riservata alle immagini di CD e DVD.

Le differenze di comportamento (specialmente rispetto all'impiego delle istantanee delle VM) fra le varie modalità appena descritte possono essere ben riassunte considerando il caso di una VM in cui sia stato installato un S.O. guest e della quale sia stata salvata un'istantanea. Immaginando di aver accidentalmente

infettato il sistema con un virus e di voler tornare alla situazione 'pulita'.

Nel caso in cui il disco sia di tipo 'normale' ripristinando l'istantanea tutto lo stato precedente del disco sarà conseguentemente ripristinato, e l'infezione da virus annullata. Con un disco di tipo 'invariabile', invece, basterà semplicemente spegnere e accendere nuovamente la VM per ritrovarla nello stato “pulito”' precedente all'infezione virale.

Al contrario, nel caso di disco 'writethrough' non sarà possibile annullare la presenza del virus mediante operazioni relative alla virtualizzazione, ma sarà necessario ripulire la macchina virtuale con un antivirus esattamente come faremmo per una macchina reale.

5.3.5 – Sezione *Audio*

La sezione successiva della finestra Impostazioni è quella relativa all'Audio; in essa è possibile decidere se la VM deve disporre di una scheda audio e se l'output audio debba essere sentito mediante l'host (cfr. *Fig.5.19*).
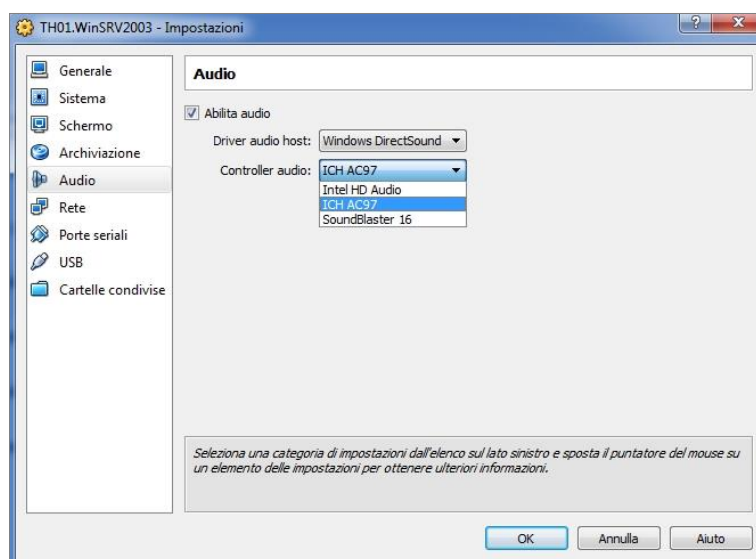
Fig. 5.19 – Impostazioni Audio

Abilitando l'audio per la VM si può scegliere tra l'emulazione di tre tipi diversi di scheda audio: Intel AC'97, Intel HD Audio controllers e Soundblaster 16; questi tre sistemi (virtuali) sono fra i più diffusi e supportati dai possibili S.O. guest.

Qualunque sia la scelta per la scheda audio virtualizzata sul Guest, è possibile selezionare quale driver audio dovrà essere utilizzato da VirtualBox per riprodurre i suoni sull'Host.

5.3.6 – Sezione *Rete*

La sezione Rete nella finestra Impostazioni permette di configurare se e in che modo VirtualBox debba provvedere a presentare una o più schede di rete virtuali alla VM, ed in che modo esse debbano eventualmente operare.
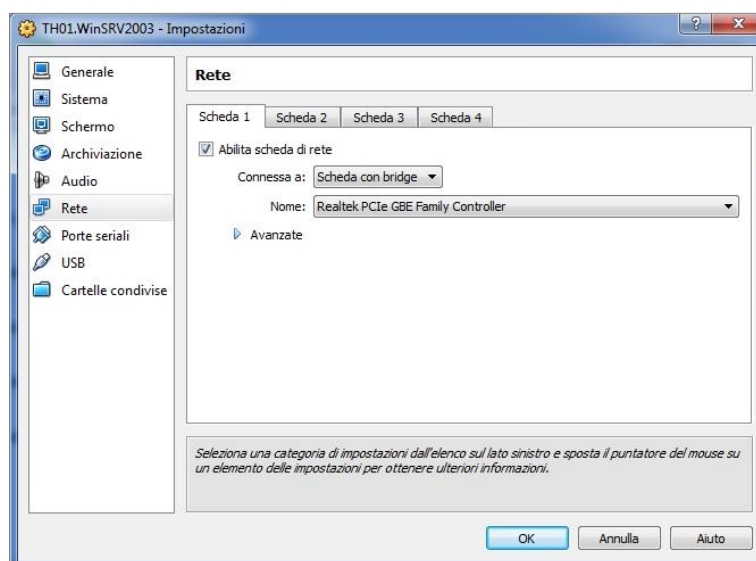
Fig. 5.20 – Impostazioni Rete

VirtualBox offre una grande flessibilità per quanto riguarda la virtualizzazione del networking, supportando fino a 8 schede di rete per VM, le prime quattro delle quali configurabili mediante l'interfaccia grafica (cfr. *Fig.5.20*), ed in generale tutte e otto configurabili mediante l'interfaccia VBoxManage a riga di comando.

Per ciascuna scheda di rete virtuale è possibile settare, individualmente, che tipo di hardware (a scelta fra sei tipi di schede fra le più comunemente supportate) debba essere virtualmente presentato al S.O. guest ed il modo in cui la scheda virtuale dovrà operare rispetto all'hardware fisico ed alla rete del server host. In particolare sono possibili i seguenti modi di operare degli adattatori di rete:

- **Non Connesso:** in questo modo il sistema guest rileva la presenza di una

scheda di rete, ma come se il cavo della ethernet virtuale fosse scollegato.

- **NAT (Network Address Translation):** è il modo più semplice di accedere ad una rete esterna da una macchina virtuale. Tipicamente non richiede alcuna configurazione aggiuntiva né nella rete dell'host né nel sistema guest, e pertanto è l'impostazione di default. Una VM con scheda direte configurata col NAT si comporta come un computer fisico connesso a Internet attraverso un router: in questo caso la funzione del router è assolta da VirtualBox che mappa il traffico da e per la VM in modo trasparente, e come se il router fosse posto tra ciascuna macchina virtuale e l'host. Questa separazione massimizza la sicurezza visto che, per default, le varie macchine virtuali presenti sull'host non parlano fra loro. Per contro si capisce che lo svantaggio della modalità NAT è che, proprio come in una rete privata dietro a un router, la VM è invisibile e irraggiungibile dall'esterno (a meno di non definire manualmente delle regole di forwarding di particolari porte).
- **Scheda con Bridge:** questa modalità risponde a necessità più avanzate come simulazioni di rete ed esecuzione di server sulla VM; con questa impostazione VirtualBox mette la scheda di rete virtuale in connessione diretta con una delle schede di rete fisiche del server host, e provvede ad intercettare e gestire direttamente lo scambio di pacchetti tra queste. In questo modo, la macchina virtuale risulta a tutti gli effetti come direttamente connessa alla medesima rete ethernet dell'host, e può essere raggiunta dall'host e da tutte le altre VM con analoga impostazione di rete.
- **Rete Interna:** questa modalità può essere usata per creare una rete virtuale (slegata da qualsiasi interfaccia di rete fisica dell'host) visibile soltanto a VM selezionate, ma non alle applicazioni in esecuzione sul server host né tanto meno al mondo esterno.
- **Scheda Solo Host:** questa modalità permette di creare una rete comprendente l'host ed un set di macchine virtuali selezionate, senza la necessità di impegnare un'interfaccia di rete fisica dell'host, ma creando piuttosto su di esso un'interfaccia di rete virtuale che fornisca connettività

fra le VM e l'host stesso.

- **Driver Generico:** questa modalità è usata raramente e realizza una generica interfaccia di rete virtuale, lasciando all'utente il compito di selezionare un driver apposito eventualmente distribuito come aggiunta di VirtualBox (è più che altro un'opzione aperta per future applicazioni e riservata agli sviluppatori).

5.3.7 – Sezione *Porte Seriali* e sezione *USB*

Le sezioni Porte Seriali e USB della finestra Impostazioni permettono di abilitare, rispettivamente, il supporto per dispositivi seriali ed USB fisicamente connessi al server host in modo che risultino virtualmente presentati alla macchina virtuale. Da notare che, non appena il guest inizia ad utilizzare un certo dispositivo USB, questo risulta immediatamente inutilizzabile per l'host.

In particolare il supporto USB di VirtualBox è assai sofisticato, permettendo non solo al sistema guest di vedere come direttamente collegati i dispositivi USB dell'host, ma anche -nel caso di accesso remoto tramite VRDP- i dispositivi USB connessi alla macchina sulla quale è in esecuzione il client RDP.

5.3.8 – Sezione *Cartelle Condivise*

La sezione Cartelle Condivise, infine, permette di definire delle cartelle nel disco fisico del server host, successivamente mappabili come cartelle condivise anche per il S.O. guest (cfr. *Fig.5.21*).

Questa impostazione, anche se riduce l'isolamento delle VM, può risultare utile per scambiare in modo semplice file fra VM e host. Il funzionamento di questa caratteristica è condizionato all'installazione nel S.O. guest delle *Guest Addition* (cfr. *paragrafo 5.1*).
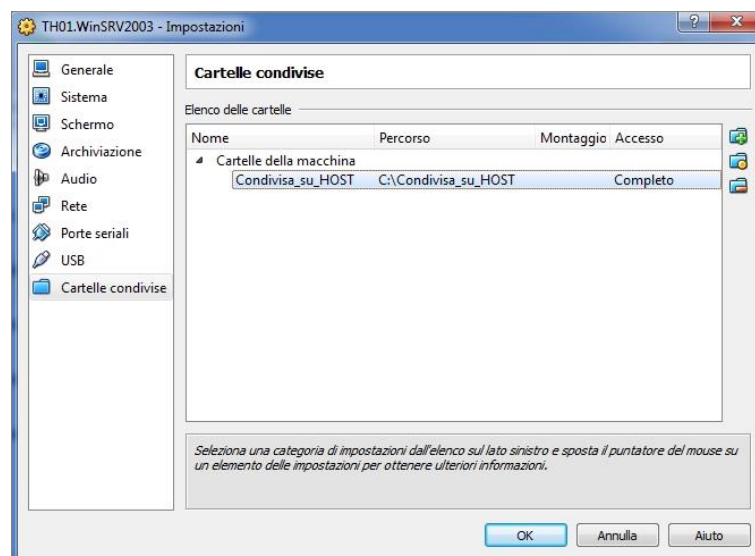
Fig. 5.21 – Impostazioni Cartelle Condivise

5.4 - L'INTERFACCIA DI VBOXMANAGE

Come già accennato nel *paragrafo 5.2.7 (Utilizzo di front-end alternativi al VirtualBox Manager)* VBoxManage è l'interfaccia a riga di comando di VirtualBox. Con essa è possibile controllare minuziosamente, sia i comportamenti del virtualizzatore che quelli delle macchine virtualizzate, dalla riga di comando del Sistema Operativo del server Host.

VBoxManage dà accesso non solo a tutte le impostazioni ed i comandi fin qui descritti, ma anche a tutta una serie di caratteristiche avanzate del motore di virtualizzazione, non altrimenti gestibili dall'ambiente grafico del Gestore di VirtualBox.

È facile immaginare che l'utilizzo di VBoxManage risulterà più ostico e decisamente meno intuitivo di quello dell'interfaccia grafica tanto più che i comandi e le regolazioni possibili sono decisamente numerosi; una trattazione completa ed approfondita di tali comandi esula ovviamente dagli intenti di questo seminario, e per essa si rimanda piuttosto al manuale del virtualizzatore VirtualBox.

Quello che invece ci interessa in questa sede, perché utile alla comprensione di alcuni strumenti che utilizzeremo nella parte operativa o comunque ai quali faremo riferimento nel presentare alcune caratteristiche di VirtualBox, è semplicemente dare alcune nozioni fondamentali relative al front-end a riga di comando di VBoxManage.
Ci sono due punti fondamentali da tenere presenti per l'utilizzo di VBoxManage: il primo è che VBoxManage deve sempre essere usato (nella shell di comando del S.O. Host, cfr. *Fig.5.22*) in abbinamento con un particolare "sottocomando" che specifica l'azione vera e propria da eseguire (ad esempio *"list"*, per elencare tutte le VM registrate, oppure *"createvm"*, o ancora *"startvm"*, rispettivamente per creare o avviare una macchina virtuale); il secondo punto è che la maggior parte dei sottocomandi richiede inoltre il passaggio di tutta una serie di parametri (ad esempio il nome della VM su cui agire) per definire il dettaglio delle operazioni o regolazioni da effettuare.
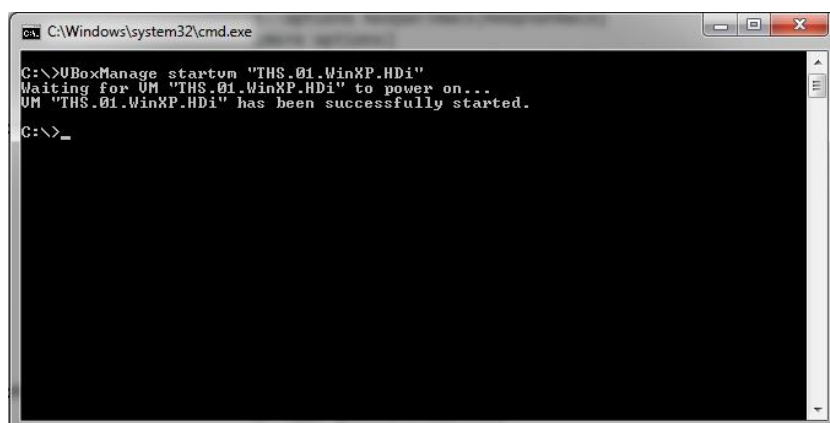
Fig. 5.22 - Utilizzo di VBoxManage da shell di comando del S.O. Host

Ad esempio il comando:

```
VBoxManage startvm "THS.01.WinXP.HDi"
```

determina l'avvio della VM di nome *THS.01.WinXP.HDi*, esattamente come se avessimo aperto il Gestore di VirtualBox ed avessimo scelto ed avviato, nella lista delle VM, la medesima macchina virtuale (cfr. *Fig.5.2*, *paragrafo 5.2.1*).

Un esempio di comando nel quale si passa più di un parametro è:

```
VBoxManage modifyvm "THS.01.WinXP.HDi" --memory "512MB" –vram “128MB”
```

con il quale si imposta, per la VM di nome *THS.01.WinXP.HDi*, un quantitativo di RAM di 512 MB e una memoria video (da presentare al S.O. guest come presente a bordo della scheda video virtuale) di 128 MB.

Un modo semplice per ottenere l’elenco di tutti i possibili sottocomandi e parametri utilizzabili con VBoxManage, è quello di impartire tale comando nella shell del S.O. Host senza nessuna ulteriore specificazione; l’output che si ottiene in tal caso è appunto una lista che costituisce una sorta di help del front-end VBoxManage (cfr. *Appendice A*).

5.5 - MACCHINE VIRTUALI REMOTE

VirtualBox permette di mostrare in remoto le VM in esecuzione sul server Host; questo significa che una VM può essere in esecuzione su un server, ma essere tuttavia mostrata su un altro computer (collegato in rete), e controllata da lì esattamente come se fosse in esecuzione su di esso.
Più in particolare VirtualBox, con l’installazione di un apposito pacchetto di estensioni rilasciato da Oracle, offre pieno supporto per il cosiddetto VRDP (VirtualBox Remote Display Protocol) una sorta di estensione -comunque retrocompatibile- del protocollo di accesso remoto di Microsoft (RDP); in sostanza quando un client *si connette*, come si usa dire, *in RDP*, (ovvero per mezzo di un software visualizzatore che sfrutta il protocollo RDP), ad una VM, si ha che gli aggiornamenti del video e l’audio sono inviati dalla macchina remota al client, mentre, per contro, gli input di mouse e tastiera sono trasferiti dal client alla VM stessa.
Il server VRDP che permette la gestione remota di una VM è tipicamente disabilitato per default sulle macchine virtuali; esso può essere facilmente abilitato

per una data VM o dall'interfaccia grafica del Gestore di VirtualBox (dalla sezione Schermo della finestra Impostazioni della macchina virtuale stessa, cfr. *paragrafo 5.3.3*, *Fig.5.17*) oppure con il seguente comando di VBoxManage:

```
VBoxManage modifyvm "THS.01.WinXP.HDi" --vrde on --vrdeport 5001
```

Poiché come già anticipato il VRDP è retrocompatibile con l'RDP di Microsoft, sarà possibile utilizzare qualsiasi viewer RDP standard per connettersi ad una tale VM (ad esempio per client Windows il comune *mstsc.exe* presente fra gli accessori standard) specificando come indirizzo IP quello del Server Host di VirtualBox (e non della VM!), e come porta quella impostata per il server VRDP stesso (nel comando dell'esempio sopra, la porta 5001).

5.5.1 – VBoxHeadless, il server di desktop remoto di VirtualBox

Sebbene qualsiasi macchina virtuale avviata dal Gestore di VirtualBox (cfr. *paragrafo 5.2.3*) ovvero con l'apposito comando *VBoxManage startvm "Nome VM"* (cfr. *paragrafo 5.4*) possa poi essere gestita anche da remoto, purché il server VRDP sia per essa abilitato, si capisce che la sua esecuzione con tutta l'interfaccia grafica completa rappresenta uno spreco di risorse nel caso in cui non si sia mai interessati a gestirla direttamente sul server host.

In altri termini se un server host ha come unico scopo quello di ospitare macchine virtuali destinate ad essere tutte amministrate esclusivamente in remoto, sarà un'inutile spreco di risorse mostrare sul server stesso un'interfaccia grafica di interazione utente-VM.

Proprio per adattarsi a tali casi (riducendo al minimo indispensabile l'impatto sulle risorse dell'Host) VirtualBox dispone di un ulteriore front-end, oltre a quelli visti, che va sotto il nome di *VBoxHeadless* e che non produce alcun output visibile sul server Host, ma si occupa soltanto di rendere disponibile via RDP la VM in questione.

Per avviare con tale front-end una macchina virtuale basta digitare, in una shell del S.O. Host, il comando:

```
VBoxHeadless --startvm "THS.01.WinXP.HDi"
```

Quando una VM viene eseguita con VBoxHeadless, non essendoci alcun altra possibilità di output, il server VRDP della stessa risulta sempre automaticamente abilitato, a prescindere dalle opzioni della VM in questione.

Si noti infine che, in una VM con sistema operativo guest che supporta l'accesso remoto, (come ad esempio, tipicamente, Windows) si potrebbe pensare di operare un'amministrazione da remoto della macchina stessa direttamente abilitando il server RDP del S.O. guest.
Tuttavia rispetto a tale soluzione, l'utilizzo del server VRDP ha il vantaggio di sfruttare un componente che sta a livello del virtualizzatore e non della VM; questo in linea di principio permette una gestione remota anche di VM i cui S.O. guest non prevedano in alcun modo il protocollo RDP.

5.5.2 – Teleporting

VirtualBox supporta una caratteristica assai interessante e tipicamente appannaggio di virtualizzatori commerciali più blasonati e soprattutto più costosi, come VMware, ovvero il cosiddetto *Teleporting*.
Con tale termine si intende la possibilità di migrare, via rete, una VM da un server host di VirtualBox ad un altro, senza interrompere il funzionamento della macchina virtuale stessa.
Il Teleporting richiede che la VM da trasferire sia in esecuzione su un server host che prende il nome di "sorgente"; l'host sul quale si vuole spostare la VM prende invece il nome di "destinazione", e la VM in esecuzione su di esso deve essere opportunamente configurata e posta in una condizione di attesa nella quale aspetta di essere contattata dall'host sorgente. Quando ciò avviene lo stato in esecuzione della VM viene trasferito dalla sorgente alla destinazione con il minimo downtime possibile.
Il Teleporting funziona su qualsiasi rete TCP/IP; sorgente e destinazione devono solo potersi scambiare gli opportuni comandi di sincronizzazione delle operazioni su una porta TCP/IP opportunamente specificata nei settaggi del teleporting

stesso.

Naturalmente, affinché la migrazione a caldo di una macchina virtuale da un host all'altro possa avvenire con successo ci sono alcuni requisiti base da rispettare, ovvero:

- La VM sull'host destinazione deve avere esattamente la medesima configurazione (per quanto riguarda le caratteristiche hardware virtualizzate) di quella sull'host sorgente che si intende trasferire a caldo.
  Questo non riguarda ovviamente le opzioni puramente descrittive, come ad esempio il nome della macchina virtuale, ma piuttosto caratteristiche come l'ammontare di memoria RAM e video presentate alla VM stessa da VirtualBox, o ancora il tipo di controller disco virtualizzato, ecc.
- Le due macchine virtuali -quella sull'host sorgente e quella sull'host destinazione- devono condividere lo stesso disco (cfr. *paragrafo 3.3.4*); ciò significa che questo deve essere fisicamente posto su un supporto di memorizzazione accessibile ad entrambe gli host: il caso più tipico può ad esempio essere quello di un NAS o in generale di qualsiasi disco raggiungibile via rete.

Se queste condizioni sono rispettate, la predisposizione del teleporting si snoda attraverso i seguenti passi:

1. Sull'host destinazione, si configura la macchina virtuale prescelta affinché questa all'avvio resti in attesa dell'arrivo della richiesta di teleporting (anziché avviarsi normalmente)
   Questo si ottiene con il seguente comando di VBoxManage:

   ```
   VBoxManage modifyvm <D_VM_name> --teleporter on --teleporterport <port>
   ```

   dove `<D_VM_name>` è il nome della macchina virtuale sull'host destinazione e `<port>` è il numero di porta TCP/IP da utilizzare sia sull'host destinazione che sul sorgente.
2. Avviando la VM così configurata sempre sull'host destinazione, questa mostra una finestra di dialogo che indica che è in attesa della richiesta di

teleporting.

3. Sull'host sorgente si avvia la VM normalmente, e, al momento in cui si vuole dare il via al teleporting si dà il seguente comando sul server host:

```
VBoxManage controlvm <S_VM_name> teleport --host <D_host> --port <port>
```

dove <S_VM_name> è il nome della VM sorgente (quella attualmente in esecuzione), è il nome o l'IP del server host di destinazione sul quale risiede la VM in attesa della richiesta di teleporting, e <port> è il numero della porta specificata -per la VM destinazione- con il comando di cui al precedente *punto 1*.

L'utilizzo del teleporting nel contesto di una infrastruttura in cui vari server, e in generale varie macchine, siano virtualizzati ed interconnessi, fra loro ed in rete, per la fornitura di servizi, permette di immaginare scenari nei quali l'operatività delle macchine virtuali può estendersi temporalmente anche oltre quella del server host fisico sulle quali le stesse sono in esecuzione.
Si può infatti immaginare di predisporre due (o più) server di host e di migrare a caldo (quindi senza l'interruzione delle attività che stanno svolgendo) le varie VM da uno all'altro, ad esempio in concomitanza di operazioni di fermo o manutenzione programmata di un server.

Fig. 5.23 - Teleporting: migrazione a caldo e bilanciamento dei carichi

Allo stesso modo si può pensare di utilizzare il teleporting per lo spostamento a caldo delle varie VM al fine di bilanciare il carico di lavoro fra più server di host, ottimizzandone così lo sfruttamento delle risorse hardware.

Esempi di schermate di VM e relativa situazione su Server Host:
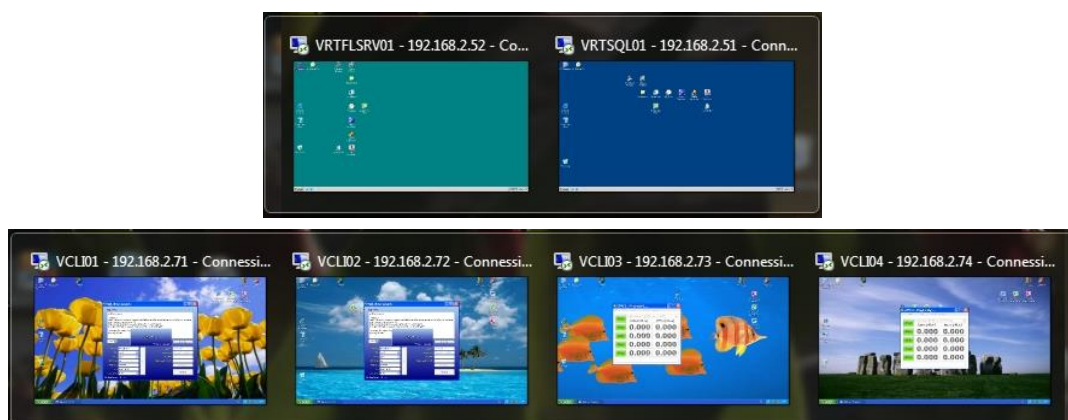
Fig. 5.24 - Miniature RDP dei 2 Server e 4 Client Virtualizzati in esecuzione contemporanea
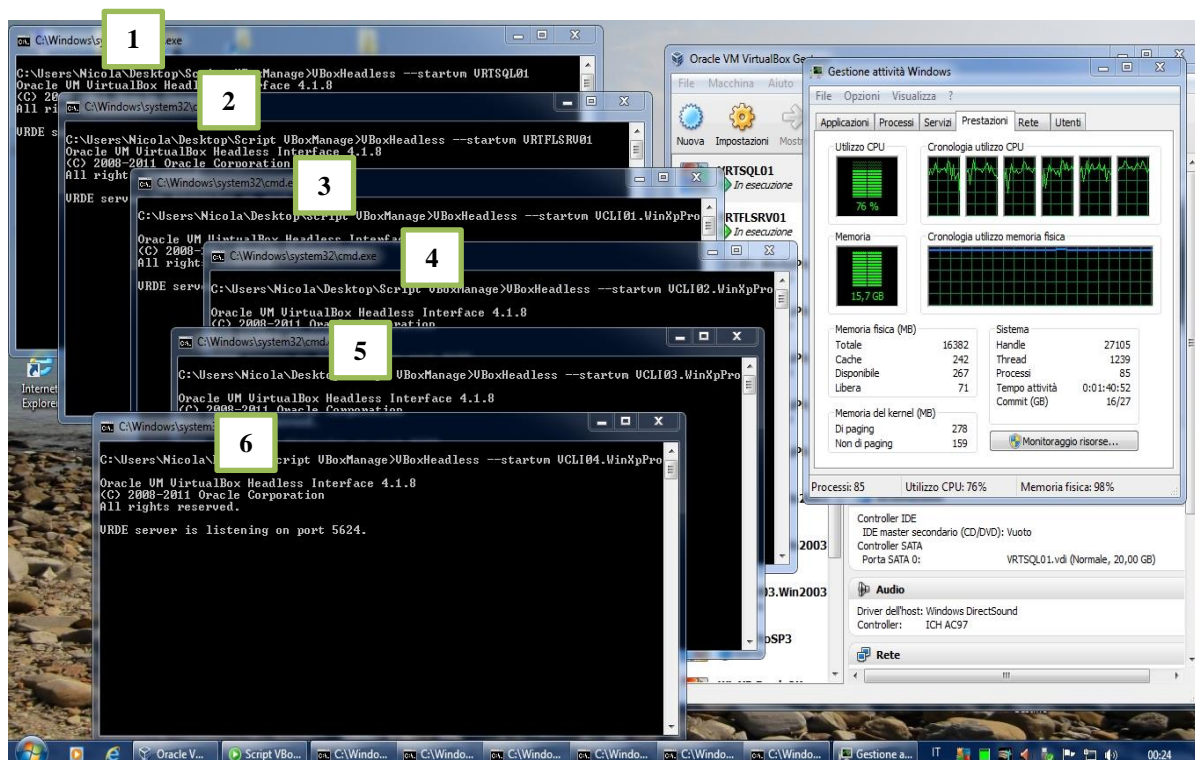
Fig. 5.25 - Print Screen Server Host VirtualBox con 6 VMs in esecuzione

5.7 - CONCLUSIONI

La Virtualizzazione si presenta, ormai da alcuni anni, come una soluzione matura in grado di offrire ad aziende ed imprese gli strumenti più efficaci per adeguarsi, nell'ottica del consolidamento e della razionalizzazione delle risorse, alle continue riconfigurazioni ed evoluzioni imposte da un ambiente (economico e tecnologico) in continua trasformazione.

Nella sua declinazione più diffusa, la Virtualizzazione è tipicamente realizzata affidandosi a blasonati software commerciali di indiscussa affidabilità in accoppiamento con risorse hardware di tutto rispetto generalmente nate e progettate proprio espressamente per l'impiego con tali software; tutto questo produce generalmente un sistema caratterizzato da costi di implementazione ed investimenti iniziali decisamente al di fuori della portata della maggioranza delle realtà medie e certamente di tutte quelle più piccole.

Ciononostante tutti i benefici della Virtualizzazione in termini di consolidamento, flessibilità di gestione, razionalizzazione, sicurezza, affidabilità e disponibilità delle risorse (per ricordare solo alcuni dei principali) sono fortemente appetibili (e talvolta addirittura determinanti per la sopravvivenza stessa) anche per la piccola e media realtà aziendale o imprenditoriale, tanto da spingere alla ricerca di paradigmi alternativi per Scenari Virtualizzati che sposino esigenze di economicità anche a costo di sacrificare qualcosa in termini di prestazioni o accettare qualche inevitabile compromesso riguardo alla misura del conseguimento dei suddetti benefici.

Il VMM VirtualBox, ovvero il software di virtualizzazione a vocazione open source sopra presentato, in abbinamento ad un hardware con caratteristiche decorose ma certamente non comparabili a quelle delle summenzionate piattaforme dedicate a prodotti commerciali di ben più alte pretese può tutto sommato ben figurare per la sostituzione con uno scenario virtualizzato di infrastrutture informatiche reali.

Appendice A

Elenco comandi di VBOXMANAGE®

Un modo semplice per ottenere l'elenco di tutti i possibili sottocomandi e parametri utilizzabili con VBoxManage, è quello di impartire tale comando nella shell del S.O. Host senza nessuna ulteriore specificazione; l'output che si ottiene in tal caso è appunto una lista di questo tipo, che costituisce una sorta di help del front-end VBoxManage [12]:

```
Usage:

VBoxManage              [-v|--version]          print version number and exit

VBoxManage              [-q|--nologo]           suppress the logo

VBoxManage list         [--long|-l]             vms|runningvms|ostypes|hostdvds|
                                                hostfloppies|bridgedifs|dhcpservers|hostinfo|
                                                hostcpuids|hddbackends|hdds|dvds|floppies|
                                                usbhost|usbfilters|systemproperties|
                                                extpacks

VBoxManage showvminfo                            <uuid>|<name> [--details]
                                                [--machinereadable]

VBoxManage showvminfo                           <uuid>|<name> --log <idx>

VBoxManage registervm                           <filename>

VBoxManage unregistervm                         <uuid>|<name> [--delete]

VBoxManage createvm                             --name <name>
                                                [--ostype <ostype>]
                                                [--register]
                                                [--basefolder <path>]
                                                [--uuid <uuid>]

VBoxManage modifyvm                             <uuid|name>
                                                [--name <name>]
                                                [--ostype <ostype>]
                                                [--memory <memorysize in MB>]
                                                [--pagefusion on|off]
                                                [--vram <vramsize in MB>]
                                                [--acpi on|off]
                                                [--ioapic on|off]
                                                [--pae on|off]
                                                [--hpet on|off]
```

```
[--hwvirtex on|off]
[--hwvirtexexcl on|off]
[--nestedpaging on|off]
[--largepages on|off]
[--vtxvpid on|off]
[--synthcpu on|off]
[--cpuidset <leaf> <eax> <ebx> <ecx> <edx>]
[--cpuidremove <leaf>]
[--cpuidremoveall]
[--hardwareuuid <uuid>]
[--cpus <number>]
[--cpuhotplug on|off]
[--plugcpu <id>]
[--unplugcpu <id>]
[--cpuexecutioncap <1-100>]
[--rtcuseutc on|off]
[--monitorcount <number>]
[--accelerate3d on|off]
[--firmware bios|efi|efi32|efi64]
[--chipset ich9|piix3]
[--bioslogofadein on|off]
[--bioslogofadeout on|off]
[--bioslogodisplaytime <msec>]
[--bioslogoimagepath <imagepath>]
[--biosbootmenu
 disabled|menuonly|messageandmenu]
[--biossystemtimeoffset <msec>]
[--biospxedebug on|off]
[--boot<1-4> none|floppy|dvd|disk|net>]
[--nic<1-N> none|null|nat|bridged|intnet|generic]
[--nictype<1-N> Am79C970A|Am79C973]
[--cableconnected<1-N> on|off]
[--nictrace<1-N> on|off]
[--nictracefile<1-N> <filename>]
[--nicproperty<1-N> name=[value]]
[--nicspeed<1-N> <kbps>]
[--nicbootprio<1-N> <priority>]
[--nicpromisc<1-N> deny|allow-vms|allow-all]
[--nicbandwidthgroup<1-N> none|<name>]
[--bridgeadapter<1-N> none|<devicename>]
[--intnet<1-N> <network name>]
[--natnet<1-N> <network>|default]
[--nicgenericdrv<1-N> <driver>
[--natsettings<1-N> [<mtu>],[<socksnd>],
[<sockrcv>],[<tcpsnd>],
[<tcprcv>]]
[--natpf<1-N> <rulename>],
tcp|udp,[<hostip>],<hostport>,[<guestip>],
<guestport>]
[--natpf<1-N> delete <rulename>]
[--nattftpprefix<1-N> <prefix>]
[--nattftpfile<1-N> <file>]
[--nattftpserver<1-N> <ip>]
[--natbindip<1-N> <ip>
[--natdnspassdomain<1-N> on|off]
[--natdnsproxy<1-N> on|off]
[--natdnshostresolver<1-N> on|off]
[--nataliasmode<1-N> default|[log],[proxyonly],
```

```
                                [sameports]]
                                [--macaddress<1-N> auto|<mac>]
                                [--mouse ps2|usb|usbtablet]
                                [--keyboard ps2|usb]
                                [--uart<1-N> off|<I/O base> <IRQ>]
                                [--uartmode<1-N> disconnected|
                                server <pipe>|client <pipe>|file <file>|
                                <devicename>]
                                [--guestmemoryballoon <balloonsize in MB>]
                                [--gueststatisticsinterval <seconds>]
                                [--audio none|null|dsound|solaudio|oss|
                                oss|coreaudio]
                                [--audiocontroller ac97|hda|sb16]
                                [--clipboard disabled|hosttoguest|guesttohost|
                                bidirectional]
                                [--vrde on|off]
                                [--vrdeextpack default|<name>
                                [--vrdeproperty <name=[value]>]
                                [--vrdeport <hostport>]
                                [--vrdeaddress <hostip>]
                                [--vrdeauthtype null|external|guest]
                                [--vrdeauthlibrary default|<name>
                                [--vrdemulticon on|off]
                                [--vrdereusecon on|off]
                                [--vrdevideochannel on|off]
                                [--vrdevideochannelquality <percent>]
                                [--usb on|off]
                                [--usbehci on|off]
                                [--snapshotfolder default|<path>]
                                [--teleporter on|off]
                                [--teleporterport <port>]
                                [--teleporteraddress <address|empty>
                                [--teleporterpassword <password>]

VBoxManage clonevm              <uuid>|<name>
                                [--snapshot <uuid>|<name>]
                                [--mode machine|machineandchildren|all]
                                [--options link|keepallmacs|keepnatmacs|
                                keepdisknames]
                                [--name <name>]
                                [--basefolder <basefolder>]
                                [--uuid <uuid>]
                                [--register]
VBoxManage import               <ovf/ova>
                                [--dry-run|-n]
                                [--options keepallmacs|keepnatmacs]
                                [more options]
                                    (run with -n to have options displayed
                                                      for a particular OVF)

VBoxManage export               <machines> --output|-o <ovf/ova>
                                [--legacy09]
                                [--manifest]
                                [--vsys <number of virtual system>]
                                [--product <product name>]
                                [--producturl <product url>]
                                [--vendor <vendor name>]
                                [--vendorurl <vendor url>]
```

```
                                          [--version <version info>]
                                          [--eula <license text>]
                                          [--eulafile <filename>]

VBoxManage startvm                        <uuid>|<name>...
                                          [--type gui|sdl|headless]

VBoxManage controlvm                      <uuid>|<name>
                                          pause|resume|reset|poweroff|savestate|
                                          acpipowerbutton|acpisleepbutton|
                                          keyboardputscancode <hex> [<hex> ...]|
                                          setlinkstate<1-N> on|off |
                                          nic<1-N> null|nat|bridged|intnet|generic
                                          [<devicename>] |
                                          nictrace<1-N> on|off
                                          nictracefile<1-N> <filename>
                                          nicproperty<1-N> name=[value]
                                          natpf<1-N> [<rulename>],tcp|udp,[<hostip>],
                                          <hostport>,[<guestip>],<guestport>
                                          natpf<1-N> delete <rulename>
                                          guestmemoryballoon <balloonsize in MB>]
                                          gueststatisticsinterval <seconds>]
                                          usbattach <uuid>|<address> |
                                          usbdetach <uuid>|<address> |
                                          vrde on|off |
                                          vrdeport <port> |
                                          vrdeproperty <name=[value]> |
                                          vrdevideochannelquality <percent>
                                          setvideomodehint <xres> <yres> <bpp> [display] |
                                          screenshotpng <file> [display] |
                                          setcredentials <username> <password> <domain>
                                          [--allowlocallogon <yes|no>] |
                                          teleport --host <name> --port <port>
                                          [--maxdowntime <msec>] [--password password]
                                          plugcpu <id>
                                          unplugcpu <id>
                                          cpuexecutioncap <1-100>

VBoxManage discardstate                   <uuid>|<name>

VBoxManage adoptstate                     <uuid>|<name> <state_file>

VBoxManage snapshot                       <uuid>|<name>
                                          take <name> [--description <desc>] [--pause] |
                                          delete <uuid>|<name> |
                                          restore <uuid>|<name> |
                                          restorecurrent |
                                          edit <uuid>|<name>|--current [--name <name>]
                                                                  [--description <desc>] |
                                          list [--details|--machinereadable]
                                          showvminfo <uuid>|<name>

VBoxManage closemedium                     disk|dvd|floppy <uuid>|<filename> [--delete]

VBoxManage storageattach                  <uuid|vmname>
                                          --storagectl <name>
                                          [--port <number>]
                                          [--device <number>]
```

```
                                [--type dvddrive|hdd|fdd]
                                [--medium none|emptydrive|
                                <uuid>|<filename>|host:<drive>|iscsi]
                                [--mtype normal|writethrough|immutable|
                                    shareable|readonly|multiattach]
                                [--comment <text>]
                                [--setuuid <uuid>]
                                [--setparentuuid <uuid>]
                                [--passthrough on|off]
                                [--tempeject on|off]
                                [--nonrotational on|off]
                                [--bandwidthgroup <name>]
                                [--forceunmount]
                                [--server <name>|<ip>]
                                [--target <target>]
                                [--tport <port>]
                                [--lun <lun>]
                                [--encodedlun <lun>]
                                [--username <username>]
                                [--password <password>]
                                [--intnet]

VBoxManage storagectl           <uuid|vmname>
                                --name <name>
                                [--add ide|sata|scsi|floppy|sas]
                                [--controller LSILogic|LSILogicSAS|BusLogic|
                                        IntelAHCI|PIIX3|PIIX4|ICH6|I82078]
                                [--sataideemulation<1-4> <1-30>]
                                [--sataportcount <1-30>]
                                [--hostiocache on|off]
                                [--bootable on|off]
                                [--remove]

VBoxManage bandwidthctl         <uuid|vmname>
                                --name <name>
                                [--add disk|network]
                                [--limit <megabytes per second>
                                [--delete]

VBoxManage showhdinfo           <uuid>|<filename>

VBoxManage createhd             --filename <filename>
                                --size <megabytes>|--sizebyte <bytes>
                                [--format VDI|VMDK|VHD] (default: VDI)
                                [--variant Standard,Fixed,Split2G,Stream,ESX]

VBoxManage modifyhd             <uuid>|<filename>
                                [--type normal|writethrough|immutable|shareable|
                                                          readonly|multiattach]
                                [--autoreset on|off]
                                [--compact]
                                [--resize <megabytes>|--resizebyte <bytes>]

VBoxManage clonehd              <uuid>|<filename> <uuid>|<outputfile>
                                [--format VDI|VMDK|VHD|RAW|<other>]
                                [--variant Standard,Fixed,Split2G,Stream,ESX]
                                [--existing]
```

```
VBoxManage convertfromraw        <filename> <outputfile>
                                 [--format VDI|VMDK|VHD]
                                 [--variant Standard,Fixed,Split2G,Stream,ESX]
                                 [--uuid <uuid>]

VBoxManage convertfromraw        stdin <outputfile> <bytes>
                                 [--format VDI|VMDK|VHD]
                                 [--variant Standard,Fixed,Split2G,Stream,ESX]
                                 [--uuid <uuid>]

VBoxManage getextradata          global|<uuid>|<name>
                                 <key>|enumerate

VBoxManage setextradata          global|<uuid>|<name>
                                 <key>
                                 [<value>] (no value deletes key)

VBoxManage setproperty           machinefolder default|<folder> |
                                 vrdeauthlibrary default|<library> |
                                 websrvauthlibrary default|null|<library> |
                                 vrdeextpack null|<library> |
                                 loghistorycount <value>

VBoxManage usbfilter             add <index,0-N>
                                 --target <uuid>|<name>|global
                                 --name <string>
                                 --action ignore|hold (global filters only)
                                 [--active yes|no] (yes)
                                 [--vendorid <XXXX>] (null)
                                 [--productid <XXXX>] (null)
                                 [--revision <IIFF>] (null)
                                 [--manufacturer <string>] (null)
                                 [--product <string>] (null)
                                 [--remote yes|no] (null, VM filters only)
                                 [--serialnumber <string>] (null)
                                 [--maskedinterfaces <XXXXXXXX>]

VBoxManage usbfilter             modify <index,0-N>
                                 --target <uuid>|<name>|global
                                 [--name <string>]
                                 [--action ignore|hold] (global filters only)
                                 [--active yes|no]
                                 [--vendorid <XXXX>|""]
                                 [--productid <XXXX>|""]
                                 [--revision <IIFF>|""]
                                 [--manufacturer <string>|""]
                                 [--product <string>|""]
                                 [--remote yes|no] (null, VM filters only)
                                 [--serialnumber <string>|""]
                                 [--maskedinterfaces <XXXXXXXX>]

VBoxManage usbfilter             remove <index,0-N>
                                 --target <uuid>|<name>|global

VBoxManage sharedfolder          add <vmname>|<uuid>
                                 --name <name> --hostpath <hostpath>
                                 [--transient] [--readonly] [--automount]
VBoxManage sharedfolder          remove <vmname>|<uuid>
```

```
                                        --name <name> [--transient]

VBoxManage debugvm                      <uuid>|<name>
                                        dumpguestcore --filename <name> |
                                        info <item> [args] | injectnmi | log [--release|
                                        --debug] <settings> ...|
                                        logdest [--release|--debug] <settings> ...|
                                        logflags [--release|--debug] <settings> ...|
                                        osdetect | osinfo | getregisters [--cpu <id>] <reg>|
                                        all ... |
                                        setregisters [--cpu <id>] <reg>=<value> ... |
                                        statistics [--reset] [--pattern <pattern>]
                                        [--descriptions]

VBoxManage metrics                      list [*|host|<vmname> [<metric_list>]]
                                                      (comma-separated)
                                        setup
                                        [--period <seconds>] (default: 1)
                                        [--samples <count>] (default: 1)
                                        [--list]
                                        [*|host|<vmname> [<metric_list>]]
                                        query [*|host|<vmname> [<metric_list>]]
                                        enable
                                        [--list]
                                        [*|host|<vmname> [<metric_list>]]
                                        disable
                                        [--list]
                                        [*|host|<vmname> [<metric_list>]]
                                        collect
                                        [--period <seconds>] (default: 1)
                                        [--samples <count>] (default: 1)
                                        [--list]
                                        [--detach]
                                        [*|host|<vmname> [<metric_list>]]

VBoxManage dhcpserver                   add|modify --netname <network_name> |
                                        [--ip <ip_address>
                                        --netmask <network_mask>
                                        --lowerip <lower_ip>
                                        --upperip <upper_ip>]
                                        [--enable | --disable]

VBoxManage dhcpserver                   remove --netname <network_name> |
VBoxManage extpack                      install [--replace] <tarball> |
                                        uninstall [--force] <name> |
                                        cleanup
```

Per ogni chiamata di VBoxManage da shell del S.O. host, può essere eseguito un solo sottocomando di quelli sopra elencati, anche se in diversi casi uno stesso sottocomando supporta svariati ulteriori sottocomandi (ciascuno con propri parametri) che in tal caso vengono eseguiti tutti nella medesima chiamata.

Bibliografia e Link

[1] ***Sistemi Operativi - Concetti Ed Esempi*** { ***A. Silberschatz***, *P.B. Galvin, G. Gagne*, 2001 ***(capp.:1,2)***}

[2] ***Storia della Virtualizzazione*** {**A. Bawcom**, P. Kenealy, W. Noonan, C. Schiller, F. Shore, A. Turriff, C. Willems, **D. Williams**: ***"Virtualization for Security"***, Syngress, 2009 ***(capp.:1,2)***}
{http://www.scribd.com/doc/75366925/Syngress-virtualization-for-Security-dec }

[3] ***Time Line della Virtualizzazione***
{http://en.wikipedia.org/wiki/Timeline_of_virtualization_development}

[4] ***Atlas Computer*** {http://en.wikipedia.org/wiki/Atlas_(computer)}

[5] ***IBM 44/44x*** {http://en.wikipedia.org/wiki/M44/44x}

[6] ***Compatible Time-Sharing System***
{http://en.wikipedia.org/wiki/Compatible_Time_Sharing_System}

[7] ***VM/370*** - R. J. Creasy, *"The origin of the VM/370 time-sharing system"*, IBM Journal of Research & Development, Vol. 25, No. 5 (September 1981)

[8] ***Cray Time Sharing System***
{http://en.wikipedia.org/wiki/Cray_Time_Sharing_System}

[9] ***Requisiti Formali Architetture Virtualizzabili*** - Popek, G. & Goldberg, R. (1974), *"Formal Requirements for Virtualizable Third Generation Architectures"*, Association for Computing Machinery, Inc.
{ http://www.dc.uba.ar/materias/so/2010/verano/descargas/articulos/VM-requirements.pdf}

[10] ***Tecnologie di Virtualizzazione: Tipi e Differenze*** - CNAPONLINE Advanced Professional Training
{http://www.cnaponline.com/articoli_virtualizzazione/40/virtualizzazione_tipi_e_differenze.htm}

[11] ***x86 Virtualization*** - TechArena Community
{http://forums.techarena.in/guides-tutorials/1104460.htm}

[12] ***Manuale Tecnico di Oracle VirtualBox***
{https://www.virtualbox.org/manual/UserManual.html}